SPaCIoS

Secure Provision and Consumption in the Internet of Services

FP7-ICT-2009-5, ICT-2009.1.4 (Trustworthy ICT)

Project No. 257876

www.spacios.eu

Deliverable D6.2.1

Industrial Service Description Languages and Security Validation Tools in IoS: Analysis, Requirements, and Advancements

Abstract

WP 6 of the SPaCIoS project aims to expedite the transferring of project results to industry, starting with the industrial partners of the consortium and later reaching the European industrial community in general. In particular, this deliverable focuses on service description languages, which not only target the technical characteristics of services but also the business and operational nature of services, and security validation tools used in industrial environments. At the same time, this deliverable will also provide inspiration to the further development of these technologies in the context of SPaCIoS.

Deliverable details

Deliverable version: *v1.0* Classification: *public*
Date of delivery: *03.10.2011* Due on: *30.09.2011*
Editors: *UNIVR, SAP and Siemens principal editors; ETH Zurich, KIT, INP and UNIGE secondary editors* Total pages: *45*

Project details

Start date: *October 01, 2010* Duration: *36 months*
Project Coordinator: *Luca Viganò*
Partners: UNIVR, ETH Zurich, KIT, INP, UNIGE, SAP, Siemens

SEVENTH FRAMEWORK PROGRAMME

(this page intentionally left blank)

Contents

List of Tables

1 Overview

This deliverable reports about our initial steps towards the transfer of the SPaCIoS results to our industrial partners and EU industry in general. In this respect we have not only studied state-of-the-art service description languages and validation tools, but also redacted a specific questionnaire intended to elicit user requirements from security analysts in industry. We have started collecting some inputs from business units at SAP and SIEMENS. All in all, this provides us with an initial, but clear picture of what industry uses in terms of description techniques for security aspects, critical security properties and security validation tools for web-based applications. These user requirements will be carefully considered in the SPaCIoS project in order to design and develop the SPaCIoS security testing platform to better support security analysts in their work. This is critical to maximize our chances of adoption of SPaCIoS techniques in industry. Usage of established standardized languages for test-cases such as TTCN-3 [8] and management of the overall test-case lifecycle are features that the SPaCIoS security testing framework could enable in order to create industrial security testing software layers offering an higher user experience to security analysts. Possibility to debug test-case execution so to offer interactivity to the security analysts is another feature we will investigate.

Of course understanding the industrial requirements and addressing them is not the only metric to be considered for adoption and usage of SPaCIoS in industry. There are other challenges that are more intrinsic to the SPaCIoS approach. The major technical challenge we will face in this industry migration is the specification of initial models for the system that we want to validate. These models capture the security-relevant aspects of the system and have to be provided to the SPaCIoS Tool to better exploit its features. An other well-known challenge is performances. While we are aware that addressing these challenges overall for any industrial domains is unlikely to happen, our strategy is to continue the approach initiated in the AVANTSSAR project where two main migration modes were implemented.

The first one, referred hereafter as the **consultancy mode**, amount to have us, the people in the SPaCIoS project, acting as formal method experts capable of capturing the system to be validated and its security requirements into formal specifications, running the validation through the SPaCIoS tool and going back to the business units with the results obtained. For instance, this consultancy mode could fit well for emerging security standards assumed to be adopted by many companies. In these situations, we can foresee a formal method expert taking care of constructing the reference formal model for the standard and to support the validation phase via the SPaCIoS tool.

The same arguments can apply for core service components of a company.

Alternatively, we plan to exploit the current trend in the IoS where model-driven development approaches are expected to play a fundamental role. New industrial business languages and concepts are emerging to target not only the technical characteristics of services (as done by, e.g., WSDL, BPEL), but also the business and operational nature of services SPaCIoS industrial partners are major players here. For instance, SAP is considerably contributing in the design of the Unified Service Description Language (USDL, [6]) to be used for its under-development Business Web market-place. State machines and similar formalisms are on the radar screen of these emerging languages to describe service behavior. For instance, it is possible to link BPMN specifications to a USDL specification of a composed service so to represent with standardized established notations service composition and orchestration. Similarly an ASLan++ specification, in which the formal model is written following an object-oriented programming style, can be attached to USDL and triggers the validation via the SPaCIoS tool. All in all, we will refer to this migration mode as **domain-specific automation**.

A variant of these two modes could have the formal method experts providing the formal model and the security analysts at the business units performing the validation via the SPaCIoS tool on their own. Hereafter this mode is referred to as **in-between** mode.

The deliverable starts with Section 2 presenting the questionnaire we redacted and the initial feedback received. Service and test description languages are are then discussed in Section 3 and Section 4 respectively. Security validation tools are outlined in Section 5. More than providing a comprehensive survey on languages and tools, we have here focused on those that are more promising with respect to the SPaCIoS migration strategy.

2 Questionnaire and Feedback from Security Analysts

2.1 Questionnaire

One of the aims of work package 6 is to capture user requirements from security analysts in industry on the use of description techniques for security aspects and security validation tools for web based applications. For this purpose, we have developed a questionnaire which is structured as follows:

1. **Technology of web-based applications.** Description of the technologies used to implement the web-based applications.
2. **Security requirements.**
   - **Security goals.** Listing of security goals relevant for security testing web applications and rating their relative importance. The following security goals were provided as a checklist (space left out for any additional goals): Authentication, Authorization, Auditing, Confidentiality, Integrity and Availability.
   - **Security vulnerabilities.** Listing of security vulnerabilities relevant for security testing of the web applications.
3. **Description of security requirements.** Techniques used to describe security requirements.
4. **Security validation tools.** Tools used to validate the security requirements.
5. **Open issues.** Any additional issues relevant for security aspects description and validation.

The complete questionnaire is given in Appendix A.

2.2 Feedback from Security Analysts

Industrial security experts were selected from SIEMENS and SAP. Since the questionnaire is concerned with sensitive information related to the security of existing or planned industrial web applications, the selection of experts was constrained by the level of trust on them. This significantly reduced the selection options. We carried out three questionnaires at SIEMENS and additional three at SAP. Industrial security experts form SIEMENS and SAP

were asked to fill out the questionnaire based on their relevant experience in real-world Web-based application development projects. The following is an aggregated result of these questionnaires.

1. **Technology of web-based applications:**

   Feedback: A variety of implementation technologies are used: Java (J2EE, JEE, Spring/Struts), PHP, AJAX, JASON, ASP.Net, SOAP, WSDL, and REST.

   Consequence: The diversity of the implementation technologies used poses a challenge for the security testing approaches to be developed in SPaCIoS, particularly for the test automation solution developed in the SPaCIoS tool.

2. **Security requirements:**

   - **Security goals:**

     Feedback: All the six security goal categories are covered by answers to the questionnaires. Among them, Authentication, Authorization, and Auditing are more prevalent and considered as more important. On the other side, Availability is rarely used as a security goal for testing.

     Consequence: Security goals Authentication, Authorization, Auditing are of high importance from user perspective and therefore should be treated more intensively in SPaCIoS. On the other side, we should not spent much effort on testing availability.

   - **Security vulnerabilities:**

     Feedback: A wide spectrum of vulnerabilities are covered. Among them, those most frequently occurring in industrial projects are: Cross-Site Scripting, Cross-Site Request Forgery, Injection, Session Management and Session Fixation, Insufficient Transport Layer Protection, and Unvalidated Redirects and Forwards.

Consequence: The SPaCIoS project should take those frequently mentioned vulnerabilities into consideration when developing and implementing technologies for security testing.

3. **Description of security requirements:**

   Feedback: Typically the description of security-related properties is provided either as informal text, which is still the most common approach to describe system requirements, or even implicitly assumed by suggesting certain design concepts for the system architecture. A systematic description of security requirements upfront of system design and implementation is still seldom. More formal description techniques than informal text are not used.

4. **Security validation tools:**

   Feedback: Various tools are used by security analysts. Source Code Analysis Tools (such as Fortify) are used to scan large code bases and locate interesting spots to investigate, when source code is available, although many false positives may be produced. Web Application Vulnerability Scanners (such as AppScan) are used to automatically scan web applications. Network Sniffers (such as Wireshark) are used to analyze network traffics. Proxy tools (such as Burp Suite) are used to analyze and manipulate HTTP messages.

   Consequence: Open source and commercial tools are used by security analysts. These tools have their own features and thus usages respectively. The purpose of SPaCIoS Tool should not be to compete with them, but rather to be complementary to them.

5. **Collected Open Issues:** The following open issues are collected from questionnaire answers. We will consider them during the execution of the SPaCIoS project.

   - Differentiation between security vulnerabilities and weaknesses: There is a lacking understanding about the different concepts, and notions that currently exist in the area of security validation. Security analysts demand for more guidelines that help them mastering their tasks.

- Testing for vulnerabilities from the outside (boundaries) (e.g., black-box testing): State-of-practice is to apply a bunch of penetration testing tools on a given web application since each tool has its strengths and weaknesses. What tools exactly shall be used for a given application can only be faithfully decided if the security analyst has a deep technical background and a long track of experiences in this field.
- Testing the effectiveness of the implemented vulnerability prevention mechanisms (White-box testing): Once a vulnerability is discovered it remains to detect the reason for it and to implement appropriate countermeasures. In practice however it is hard to decide if an implemented measure is a complete solution to the detected problem.
- Test planning/coordination tool is not available to better organize penetration testing activities.

3 Service Description Languages

3.1 ASLan++

Short overview

In AVANTSSAR project, we spent significant effort to devise a formal language for security-relevant aspects of service and service composition. This resulted in ASLan++ [4].

ASLan++ is a formal language for specifying security-sensitive service-oriented architectures, their associated security policies, and their trust and security properties.

We have developed ASLan++ to achieve the following design goals:

- The language should be expressive enough to model a wide range of service-oriented architectures while allowing for succinct specifications.
- The language should facilitate the specification of services at a high abstraction level in order to reduce model complexity as much as possible.
- The language should be close to specification languages for security protocols and web services, but also to procedural and object-oriented programming languages, so that it can be employed by users who are not experts of formal protocol/service specification languages.

In order to support formal specification of static and dynamic services and policy composition, ASLan++ introduces a number of features.

- Control flow constructs (e.g., if and while) enhance the readability and conciseness of the specifications, and make the specification job easier for modelers who are already familiar with programming languages.
- Modularity is supported by the use of entities. Each entity is specified separately and can then be instantiated multiple times and composed with others. This allows the specifier, in particular, to localize policies in each entity by clarifying, for instance, who is responsible to grant or deny authorization as well as the various trust relationships between entities.
- There is an intuitive notation for channels, which may be used both as assumptions and as service goals and provides a simple but powerful way to specify communication and service compositionality.

Future Development Plan

As stated before, ASLan++ was designed with the goal to make it usable by users who are not experts of formal protocol or service specification languages. In particular its object-oriented programming style should allow people from industry to specify their own ASLan++ specifications capturing their system features. Of course this requires some education and start-up time.

At the same time, chances for adoption of ASLan++ could be significantly increased by creating tools and add-ons for helping industrial people in writing ASLan++ specifications. Examples of these tools could be editors helping user to write syntactically correct ASLan++ specification, and debugging tool simulating the system behaviors and helping user to specify system behaviors as intended. It will be better if these tools could be integrated into the development environment familiar to industrial people.

Last but not least, model inference techniques investigated in WP2 could provide an effective way to automatically infer or adjust the ASLan++ model for a running system, or at least provide a basis of the ASLan++ model from which the user could improve to include more behaviors. In order to achieve this goal, in addition to having an efficient model inference and adjustment algorithm, test drivers need to be developed to connect the model inference engine and the running system in industrial environment. Of course we are aware that model inference is a very challenging topic, but successful results there, even in a few industrial domains, could be of great values for the industry migration of SPaCIoS technologies and tools.

As aforementioned, there are several possible mode of industrial migration of SPaCIoS: consultancy mode, domain-specific automation mode, and the in-between mode.

In our industry migration we will evaluate with our business units the feasibility and added value (if any) of these ideas towards the domain-specific automation mode.

Besides this, ASLan++ can be a key factor towards the consultancy migration mode. ASLan++ allows people familiar with SPaCIoS to quickly write the formal specification so to trigger and instrument the SPaCIoS methodology and tool. For example, SIEMENS people involved in SPaCIoS could capture key features of eHealth system in ASLan++ quickly, run SPaCIoS tools based on the specification, and discuss the outcome with the business units developing the eHealth system. By doing this, the formal method and security experts could get more insight of the system, and the business units could improve the system according to results of formal analysis.

Moreover, the in-between variant of our strategy for industry migration could also leverage on ASLan++. For instance, for those security standard destined to large adoption, we can foresee the usage of ASLan++ to formally capture the reference implementation. Vendors and industrial companies adopting the standard could then use SPaCIoS to generate and run test cases on the specific deployment of that security standard in their premises. Of course this requires some adjustment of the reference model into the one capturing the specific deployment. This can be done manually by changing the reference model. Since ASLan++ was designed to have a style similar to mainstream programming languages, this adjustment is not difficult for non-expert of formal method. We also believe this adjustment can be done at some extent automatically by means of SPaCIoS tool: e.g., the model inference/refinement module of SPaCIoS could observe the message exchange in the real deployment and to adjust the reference model accordingly.

3.2 Unified Service Description Languages (USDL)

Overview of USDL

The Unified Service Description Language (USDL) is proposed as a "master data model for services" to describe various types of services ranging from professional to electronic services. It aims at a holistic service description putting a special focus on business and operational aspects such as ownership and provisioning, release stages in a service network, composition and bundling, pricing and legal aspects among others, in addition to technical aspects.

First iterations of USDL were solely built by SAP Research expending several person years of effort. About a dozen researchers at SAP Research contributed to the model by bringing in their expertise from different backgrounds (computer scientists, incl. security and SLA experts, business economists, legal scientists, etc.). This was carried out in the context of several publicly funded research projects under the Internet of Services theme. The current specifications of this work stream can be found at [6] and is known as USDL3.

Subsequent iterations of USDL include the contributions and evaluation feedbacks of partners external to SAP Research. As an example, the German Fraunhofer FOKUS institute is prompted to include aspects such as identity management and Siemens evaluates USDL in controlled experiments in their setting. Finally, the scope of input is broadened even wider by approaching a standardization body starting with W3C Incubator group [9]. The incubator group was launched in September 2011 with four initiating members

including SAP and SIEMENS.

USDL Modules

The distinction in business, operational, and technical information carries the main idea of USDL, namely the unification and interconnection of service information from all areas of the spectrum. Yet the distinction in business, operational, and technical aspects proved to be too coarse-grained for an adequate structuring. Instead, USDL is split into several packages (according to UML terminology) as a response to the requirement of modularity. Each package represents one module and contains one class model. The resulting modules are briefly explained as follows.

Foundation Module captures concepts that are common among several aspects, e.g., concepts of naming and identification, or concepts that are completely independent of "service," e.g., organizations or persons.

Service Level Module captures concepts concerned with guarantees regarding quality of service operation, which are claimed/requested by different actors involved in the provisioning, delivery and consumption of a service.

Participants Module captures concepts related to the actors that participate in the provisioning, delivery and consumption of services, e.g., provider, intermediary, stakeholder and consumer.

Pricing Module captures concepts that explicate the pricing structure of a service, e.g., price plan, price component and price level.

Legal Module captures licenses and copy rights according to laws.

Service Module captures central service concepts, e.g., service and service bundle, and their relation to other service description aspects.

Interaction Module captures concepts that outline the sequence(s) of interactions between a consumer and a service (respectively the actors involved in delivery) -necessary to successfully complete service execution.

Functional Module captures concepts that describe the functionality offered as a service, e.g., function, parameter and fault.

Technical Module captures concepts that describe available means to access a service, e.g., interface and access protocols.

In the following, we provide more detailed information about Service Module and Service Level Module, which provide service composition and security goals information. This information is more relevant for SPaCIoS industrial migration activities we are envisaging for the time being. Note that other modules could be relevant too. For example, the information about in what order individual functions of a service have to be performed provided in the Interaction Module could be related to behavior description in an ASLan++ model of the service, and the function and interface information provided in Functional and Technical Modules could be essential to help connecting the SPaCIoS Tool (test drivers) to the actual service.

Service and Service Composition Description in Service Module

The Service Module can be regarded as the center of USDL. It ties together all aspects of service description distributed across the USDL modules. At the heart of the Service Module are concepts that represent services provisioned into service networks (e.g., services, composite service, and service bundles).

The basic definition of notion of **service** is that of a distinct set of capabilities intended to be rendered to a customer, partially or as a whole. Capabilities, in turn, may be reused or aggregated to provide more complex and comprehensive capabilities. Services assembled in such a way are a special type called **composite service** in USDL. Concrete composition techniques employed nowadays range from traditional distributed software development to process-based composite application design, and recently mash-up like composition of widgets. From this definition stems the fact that some or all of the parts of a composite service are itself available as services in a service network. They are, thus, rendered by providers, which are different from the provider of the composite. Another inherent characteristic of composite services is that their parts manifest functional dependencies among each other. Moreover, the capabilities or functionalities offered by a composite are of a higher order and cannot simply be recreated by rendering the individual parts independently.

Technically, in addition to some attributes, the class **service** has a relation **additionalDocumentation** which is a set of optional links to additional (external) material that gives further description of the service, service demonstrations, reviews, etc. Also, the class **CompositeService** has a relation **parts** describing the set of parts that are reused or assembled into the composite, and a relation **implementationSpecification** which is a reference to the formal specification of the composition, e.g., orchestration

in BPEL.

Security Description in Service Level Module

Service Level Agreements (SLAs) are a common way to formally specify such functional and nonfunctional conditions under which services are or are to be delivered. However, SLAs in practice are specified at the top-level interface between a service provider and a service customer only. Customers and providers can use top-level SLAs to monitor whether the actual service delivery complies with the agreed SLA terms. In case of SLA violations, penalties or compensations can be directly derived. The USDL Service Level Module allows modeling such information including security using the following classes.

Class serviceLevelProfile represents a set of service level specifications that are combined into one profile and which are offered/negotiated/agreed as a whole. For each **service**, a set of **serviceLevelProfiles** could be specified.

Class serviceLevel specifies a single service level objective as it characterizes an offered, negotiated or agreed service. Service levels are defined by the parties participating in service provisioning, delivery and consumption, and express assertions that are claimed or expected to hold during these activities. A relation **obligatedParty** of service level specified the party that is in charge to guarantee/enforce this service level.

Class serviceLevelExpression specifies an expression that is evaluated in the context of a service level state or action. For this purpose it may reference a set of service level attributes (constants, metrics or variable references) and define relationships between these attributes, e.g., Boolean or arithmetic operands.

Class securityAttribute is a kind of service level attribute that describes guarantees about security in an abstract way. It has three attributes, **securityGoals** which could take one of the values integrity, confidentiality, identification, authentication, authorization, privacy and accountability, **realizationLevels** which specifies at what layer in the communication stack the security element is targeted at and could be one of the values transport, message, application and session, and **requirementLevel** which indicates the desired implementation degree of a security goal and could be one of the values none, low, medium and high. These abstract requirements are then

interpreted in the context of the particular platform that provides access to the service, using a "platform-specific security profile".

Class securityMetric is another kind of service level attribute that describes guarantees about security. It also specifies **securityGoals** and **realizationLevels**. Unlike **securityAttribute**, it does not express requirement level, but defines more concrete security properties with link to technical artifacts. A service provider can then specify some claims in terms of mechanism, such as internal procedure to erase Personally Identifiable Information after a certain amount of time to cover a privacy SecurityGoal.

Future Development Plan

USDL provides a unified and extensible way to describe various kinds of services which could be provisioned, deployed, aggregated, and consumed. By relating various artifacts created using SPaCIoS techniques to various modules of USDL, the providers, aggregators, and consumers could take advantage of formal facilitates developed in SPaCIoS while keep the normal way using USDL. Some of the possible directions of relating SPaCIoS artifacts to USDL are discussed as follows.

- For a single service, provide a model in ASLan++ specification in **additionalDocumentation**, and relate expected security goals specified in **securityAttribute** or **securityMetric** to security goals expressed in ASLan++. With these information, test cases could be generated using SPaCIoS tools to check whether the expected security goals are implemented properly in the service. Service intermediaries, service aggregators, and service consumers could execute the test cases to obtain certain level of confidence on the quality of the service with respect to security.

  Relating informal security goals in USDL to formal security goals in ASLan++ could be done by formal method experts, and this information could be reused in multiple services. For the service provider, providing a model of a service needs some additional effort. This effort is compensated by the fact that with a higher confidence about the quality, the service has likely higher chance to be selected and consumed.

- For a composite service, provide an orchestration specification such as BPEL in **implementationSpecification**, provide security goals and security assumptions in **securityAttribute** or **securityMetric**. At

the same time, for each service described in **parts** of the composite service, provide security goals and security assumptions in **securityAttribute** or **securityMetric**. With these information, SPaCIoS tools could be used to validate whether the security goals of the composite service could be satisfied in such a composition, and to generate test cases to check whether the expected security goals are reached in the implementations of all the services. Service intermediaries, service aggregators, and service consumers could all benefit from this validation and testing.

In order to achieve the benefit just described, orchestration specification needs to be related to formal model in ASLan++. In SAP's industrial migration of AVANTSSAR technologies, BPMN specification was translated to HLPSL++. This work could be continued using ASLan++ as the target language.

- In both cases just described, the test case generated could be "stored" in **additionalDocumentation** of the service. In this way, whenever needed by a participants in the service marketplace, test cases could be executed using SPaCIoS tools to check the implementation of the service.

More investigation and experimentation are needed for each possible direction aforementioned. We will take case studies from the USDL community to relate to and take advantage of SPaCIoS technologies.

3.3 Future Development Plan

We already described the future development plan for ASLan++ and USDL in the respective sections. Here we summarize the overall development plan in Table 1.

Table 1: Service Description Language and Migration Mode

	ASLan++	USDL
Consultancy Mode	Key factor. Used by experts of formal method.	Not closely related
Domain-Specific Automation Mode	Used by industrial people. Tools such as editor and debugger could be helpful	Information contained in USDL specification, such as security properties and service orchestration, need to be related to formal model elements in ASLan++. Thus, SPaCIoS tool could be used to validate security features of services
In-Between Mode	Used by industrial people to adjust reference model. Model inference and adjustment tools could be helpful	Similar to the case of "Domain-Specific Automation Mode", relationship needs to be established between information in USDL and formal model

4 Test Description Languages

4.1 TTCN-3

TTCN-3 (Testing and Test Control Notation version 3), which is an ETSI standard, is a flexible and powerful language applicable to the specification of all types of reactive system tests over a variety of communication interfaces. Typical areas of application are protocol testing (including mobile and Internet protocols), service testing (including supplementary services), module testing, testing of CORBA based platforms, API testing, etc. TTCN-3 is not restricted to conformance testing and can be used for many other kinds of testing including interoperability, robustness, regression, system and integration testing.

One main feature of TTCN-3 is the separation of concern between Abstract Test Suites and the Adapter Layer which allows full portability of test suites and thus make them independent of any platform implementation. The test adapter handles all platform and implementation languages (java, C, C++) issues for the communication with a System Under Test and also the actual coding and decoding requirements of an application.

In addition to that, according to the standard document, TTCN-3 includes the following essential characteristics:

- the ability to specify dynamic concurrent testing configurations;
- operations for procedure-based and message-based communication;
- the ability to specify encoding information and other attributes (including user extensibility);
- the ability to specify data and signature templates with powerful matching mechanisms;
- value parameterization;
- the assignment and handling of test verdicts;
- test suite parameterization and test case selection mechanisms;
- combined use of TTCN-3 with other languages;
- well-defined syntax, interchange format and static semantics;
- different presentation formats (e.g., tabular and graphical presentation formats);

- a precise execution algorithm (operational semantics).

The top-level unit of TTCN-3 is a module. A module can import definitions from other modules. Modules can have module parameters to allow test suite parameterization. A module consists of a definitions part and a control part. The definitions part of a module defines test components, communication ports, data types, constants, test data templates, functions, signatures for procedure calls at ports, test cases, etc. The control part of a module calls the test cases and controls their execution. The control part may also declare (local) variables, etc. Program statements (such as if-else and do-while) can be used to specify the selection and execution order of individual test cases.

Dynamic test behavior is expressed as test cases. TTCN-3 program statements include powerful behavior description mechanisms such as alternative reception of communication and timer events, interleaving and default behavior. Test verdict assignment and logging mechanisms are also supported.

An example of test case is as follows:

```
testcase Test_Case_001() runs on Component_A
  system configuration_01 {
    activate(Default_1("0"));
    map(self:PCO, system:PCO);
    PCO.send(Invite_s_1);
    T1.start;
    PCO.receive(Response_r_1);
    setverdict(pass);
    T1.stop;
    postamble("0");
    stop;
}
```

An example of the control part is as follows:

```
control{
  var integer count;
  if(execute(Test_Case_001()) == pass) {
    // Execute test case 10 times
    count := 0;
    while( count <= 10) {
      execute(Test_Case_002());
      count := count + 1;
    } // end while
  } // end if
} // end control
```

Test cases specified in TTCN-3 are executed in a TTCN-3 Test System. A TTCN-3 test system can be thought of conceptually as a set of interacting entities where each entity corresponds to a particular aspect of functionality in a test system implementation. These entities manage test execution, interpreting or executing compiled TTCN-3 code, realize proper communication with the SUT, implement external functions, and handle timer operations.

4.2 eCATT of SAP

eCATT , the Extended Computer Aided Test Tool [7], is an automated testing tool that allows users to create automated test cases for the majority of applications running in SAP GUI for Windows/Java/HTML or Web Dynpro environments. Like other test tools, it works by making a recording of an application, which users can then parameterize and replay with differing sets of input values. Users can test the behavior of the application by reading and testing the values returned by the application.

eCATT differs from external tools in that it provides full access to the application server and database layers of the system, allowing users to test function modules, BAPIs as well as Web Services, perform checks against the database, and interrogate or simulate changes to customizing settings.

Process eCATT provides an environment for developing tests. Rather than creating a single object that defines every aspect of a test, eCATT has four separate object types. The first three form the building blocks of a test, and the fourth combines the others into a complete test case. The following paragraphs summarizes the development process and how the different eCATT objects fit into that process.

1. Define what you want to test within the scope of your project and make sure that the test system and the systems to be tested are prepared for use with eCATT.

2. Create a **System Data Container** in which you map out the system landscape for the project. Without a system data container, you cannot write test scripts that access other systems.

3. Create the **Test Scripts** themselves.

4. Consider the data that you will need to run the tests and arrange it in **Test Data Containers** to allow the maximum degree of reuse and to eliminate as much redundancy as possible.

5. Assemble the **Test Configurations** from the other eCATT objects.

6. Test configurations can be assigned to test catalogs and test plans within the Test Workbench. The configurations can then be assigned to individual users for testing.

An executed test configuration produces its results in the form of a log. Not only does the log provide a simple *Pass* or *Fail* result for the complete test, it also provides a permanent and detailed record of the test.

A test script consists of three principal parts: its attributes, the script commands, and the parameters.

The test script has mandatory **attributes** (title, package, person responsible, and application component) as well as attributes containing administrative information. Two important attributes are the maintenance system and the versioning information. You need to assign a system data container in the maintenance attributes to enable the test script to address the system landscape during development. A test script can exist in several versions and the validity of a test script for testing a given system, is determined by the versioning information.

The import and export **parameters** define the interface of the test script so that values can be passed to and from the script. You can also create local variables that are only used within the test script.

The **commands** describe the test. Typically, a script contains one or more recorded transactions with the associated checks and calculations but it need not do so. For example, it could contain just some usefully functionality that can be referenced from another script, or it could contain a series of references to other scripts to build a more complex test out of reusable units.

The test scripts could be downloaded or uploaded as XML files.

4.3 Future Development Plan

With SPaCIoS technologies including model checking and test case generation, test cases are generated to probe the system with the ultimate purpose of checking whether certain security goals are achieved by a specific implementation of a system. On the other side, test description languages and their supporting systems, such as TTCN-3, are used widely in industrial environment to specify and execute test cases. Based on such an observation, a future development plan is to translate the test cases generated by SPaCIoS tools to existing test description languages such as TTCN-3 or test scripts used by eCATT.

The following benefits can be obtained by doing this translation:

- Test cases will be executed in existing industrial environments without introducing another test execution engine into the daily life of security testing practitioners. The cost of introducing the SPaCIoS technologies into existing product life cycle will be lower.
- Test cases specified in a standard language such as TTCN-3 are platform independent. The test adapter handles all platforms and implementation languages (java, C, C++) issues for the communication with a System Under Test and also the actual coding and decoding requirements of an application. In this way, the test case specification could be relatively abstract and easy to understand and maintain.
- Test cases could be generated and stored in a library. In this way, regression testing of system or service implementations can be supported as well as test-case lifecycle management

In order to achieve these potential benefits, test case translators to different target test case specification languages will be developed, according to the requirements of migrating SPaCIoS technologies to existing industrial environments.

5 Security validation Tools

5.1 Source code analyzer

Source code analysis tools are used for software validation in industry. A source code analyzer automatically inspects the source code of SUV to find vulnerabilities and faults. The inspection is often guided by a fixed set of patterns and rules that indicate possible security vulnerabilities. Therefore, a fixed set of vulnerabilities, and not all of them, can be detected using source code analyzers. Moreover, not all the reported "vulnerabilities" are in fact faults: reducing the false positive rate is a challenge in practice. Fortify and Parasoft are two examples of source code analysis tools.

- Fortify can detect around 225 types of vulnerabilities, by inspecting source codes in a number of programming languages, including C/C++, Java and .NET.
- Parasoft analyzes source codes in various programming languages including Java, C/C++ and .NET. A number of patterns and rules guide the analysis, e.g., rules for detecting input-based attacks, unsafe environment configuration, and unsafe error handling and logging.

Conclusion: SPaCIoS does not perform code analysis, as in many cases the source code of the services is not available in the validation phase. In this sense, SPaCIoS complements source code analyzers.

5.2 Port Scanners

Port scanners are very useful in a corporate scenario. They can be used by system and network administrator for tasks such as network inventory, network or services (policy) monitoring, detection of hosts and/or services.

Port scanners usually rely on Transmission Control Protocol (TCP) but some can also use:

- User Datagram Protocol (UDP),
- Stream Control Transmission Protocol (SCTP),
- File Transfer Protocol (FTP), and
- Internet Control Message Protocol (ICMP).

5.2.1 Nmap

One of the most use scanner available nowadays is Nmap (Network Mapper http://www.nmap.org/). It is a free and open source utility for network exploration or security auditing. It was designed to rapidly scan large networks, but works fine against single hosts.

Nmap features include:

- Host Discovery - Identifying hosts on a network, for example listing the hosts which respond to pings, or which have a particular port open.
- Port Scanning - Enumerating the open ports on one or more target hosts.
- Version Detection - Interrogating network services on remote devices to determine the application name and version number.
- OS Detection - Remotely determining the operating system and some hardware characteristics of network devices.
- Scriptable interaction with the target - using Nmap Scripting Engine (NSE) and Lua programming language, customized queries can be made (See http://www.nmap.org/book/man-nse.html for more details).
- In addition to these features, Nmap can provide further information on targets, including reverse DNS names, device types, and MAC addresses.

Typical uses of Nmap:

- Auditing the security of a device, by identifying the network connections which can be made to it.
- Identifying open ports on a target host in preparation for auditing.
- Network inventory, Network mapping, maintenance, and asset management.
- Auditing the security of a network, by identifying unexpected new servers.

Nmap have many type of scans; in the SPaCIoS project four of them could be used to map the SUV:

- **Ping scan**: gain a taxonomy about the SUV (it could be composed of a group of machines).

- **Syn scan**: This type of scan is very fast: it can scan thousands ports in a second over a network and it is not limited by firewall. The SYN scan is stealthy and not so invasive.

- **UDP scan**: vulnerable UDP services are very common (e.g., DNS, SNMP or DHCP), also this type of scan could be a good choice if we are dealing with Microsoft devices.

- **Connect scan**: It is the right scan if we are working with IPv6 networks.

We can use Nmap with the command:
`nmap [Scan Type(s)] [Options] {target}`
For each of the previous scans we have:

Syn: `[-sS]`

+ Never creates an application session
+ Never appears in a log (very quiet)
- Requires privileged access
- Will send a large number of RSTs (reset frames)

UDP `[-sU]`

+ Lower overhead
+ Works well on Microsoft devices
- Privileged access only
- ICMP port unreachable packets may increase the number of packets
- Many devices that use Unix/Linux OSes will limit UDP port throughput

Ping `[-sP]`

+ Very common traffic pattern
+ Very fast
- Can't be used with other scan type
- Limited information

Connect() `[-sT]`

+ No special privileges
- Appears in log files
- Uses additional resources on the remote device

In SPaCIoS the `SYN-SCAN` could be a useful instrument because:

- it is very clean,

- it does not cause any problem with the normal operation of the remote device,
- it provides only information on the ports (open, closed or filtered), and
- it works in every situation where TCP works (it uses the TCP SYN process).

As we have seen also the `PING/UDP/CONNECT-SCANS` are useful in particular situations where the `SYN-SCAN` could have some difficulties.

5.3 Vulnerability Scanners

Vulnerability scanners work from a database of documented network service security defects, trying to discover each defect by probing each available service of the target range of hosts. An example of vulnerability scanner is Nessus of Tenable Network Security®.

Tenable Nessus

Nessus features high-speed discovery, configuration auditing, asset profiling, sensitive data discovery and vulnerability analysis of security posture. Nessus scanners can be distributed throughout an entire enterprise, inside DMZs and across physically separate networks.

A Nessus "policy" consists of configuration options related to performing a vulnerability scan. These options include, but are not limited to:

- Parameters that control technical aspects of the scan such as timeouts, number of hosts, type of port scanner and more.
- Credentials for local scans (e.g., Windows, SSH), authenticated Oracle database scans, HTTP, FTP, POP, IMAP or Kerberos based authentication.
- Granular family or plugin based scan specifications.
- Database compliance policy checks, report verbosity, service detection scan settings, Unix compliance checks and more.

Nessus ships with several default policies provided by Tenable Network Security, Inc. They are provided as templates to assist in creating custom policies for an organization or to use as-is in order to start basic scans of resources.

- The **External Network Scan** policy is tuned to scan externally facing hosts, which typically present fewer services to the network. The plugins associated with known web application vulnerabilities (e.g., CGI Abuses and XSS) are enabled in this policy. Also, all 65,535 ports are scanned for on each target.
- The **Internal Network Scan** policy is tuned for better performance, taking into account that it may be used to scan large internal networks with many hosts, several exposed services, and embedded systems such as printers.
- The **Web App Tests** policy enables fuzzing capabilities in Nessus, which will cause Nessus to spider all discovered web sites and then look for vulnerabilities present in each of the parameters, including XSS, SQL, command injection and several more.
- The **Prepare for PCI DSS Audits** policy enables the built-in PCI DSS compliance checks that compare scan results with the PCI standards and produces a report on your compliance posture. Organizations preparing for a PCI DSS assessment can use this policy to prepare their network and systems for PCI DSS compliance.

5.4 Application Scanners

Here we describe the purpose and concept of application scanners. In contrast to Web Application Assessment proxies, located between browser and webserver and used in an interactive way, Application Scanners are used without a browser, interacting directly with the server in an entirely black box manner. They attempt a variety of common well-known attacks, including Cross Site Scripting (XSS), SQL Injection, Local File Inclusion, Remote Code Execution, and HTTP Response Splitting, on the different pages of the application. The advantage over Web Application Assessment proxies is that they run automatically, without human intervention (although a careful configuration is necessary to provide the scanner enough information to access the application correctly). The disadvantage of black box and non-interactive Web scanners is that they often do not recognize many important vulnerabilities present. Notice, for instance that OWASP warns against the use of black box testing for Dom-Based XSS, see [1]. But the situation is changing rapidly as new technologies are being developed. Also, Web scanners are being integrated into suites that access the source code.

5.4.1 IBM Rational AppScan

IBM Rational AppScan (formerly, Watchfire AppScan) automates web application security audits to help ensure the security and compliance of websites. It may be used for most types of web application security testing procedures - outsourced, individual scans and enterprise-wide analysis - and creates reports for application developers, quality assurance teams, penetration testers, security auditors and senior management. It is used in variety of application instances, including test, development and production

The functionality is basically the following: first the site is crawled and all linked pages in a given domain are found. Then all those pages are checked against typical attacks web forms (in particular SQL Injection, Cross Site Scripting (XSS), and Buffer Overflows). Finally, reports for the different purposes are generated.

Assessing the performance, quality, and accuracy of web application vulnerability scanners is very difficult. In January 2009 a Web Vulnerability Scanners Evaluation was published in [3]. This report is rather well-known and can be found in different pages over the Internet. In that study, AppScan 7.8 was compared to two other commercial application scanners and the author concludes: "AppScan scored worst in almost all the cases. They are finishing the scan quickly because they don't do a comprehensive test."

In a response from IBM (private communication to Siemens), the vendor challenges the results: "[We ...] analyzed and re-evaluated the comparison results, and through this process have discovered fundamental flaws in the initial evaluation process: All scanners were run without any configuration whatsoever [...] No login credentials were provided to the scanners, [...] we did discover (and validate) new vulnerabilities, that other scanners did not find in the original report [...] Some vulnerable URLs existed in URLs that any other blackbox scanner could have never located, since the application did not contain direct links to those URLs [...] We believe that comparing a non-configured blackbox scanner, to a scanner that enjoyed access to application source code and file system, is a futile experiment."

Recently, a web application security scanner comparison was published in [5]. The survey, covered 60 different open source and commercial scanners, and AppScan (v8.0.03) was rated as the best regarding scanning capabilities. For the test, the vulnerable web application *WAVSEP* was used. WAVSEP was designed precisely for the purpose of assessing the features, quality and accuracy of web application vulnerability scanners [2]. The application contains a large number of test cases:

- Reflected XSS: 66 test cases

- Error Based SQL Injection: 80 test cases
- Blind SQL Injection: 46 test cases
- Time Based SQL Injection: 10 test cases

Appscan discovered 100% of all of the reflected XSS test cases, with no false positives, and it found 127 issues (93.38% success rate) for SQL Injection tests, with 3 false positives (out of the 10 extra designed "false positive test cases", that is, situations where simple decisions may detect a vulnerability, but there is none).

In the latest version 8, AppScan has included a new technology, called JavaScript Security Analyzer (JSA), that saves, for each URL, the entire HTTP response stream. Then, JSA applies a set of JavaScript taint analysis rules and discovers data flows from source to sink that do not go through a sanitizer. In other words, although JSA performs information-flow analysis, but not in the original source code of the application (as static analyzers do), but rather on the JavaScript sent back from the server. This feature is particularly interesting for websites that use for instance AJAX, in order to generate HTML and JavaScript code on the fly.

5.5 Web Application Assessment Proxy

In this section we describe the purpose and concept of web application assessment proxies. In contrast to network sniffers, proxies are applications that analyze the communication of a distributed application. They are located between the application at the client side (e.g., browser) and the service running at the server side (e.g., webserver). For instance a web application assessment proxy can be a framework that analyzes the HTTP and HTTPS protocol of web applications. Proxies are able to intercept, analyze, and/or manipulate information exchanged between different distributed parties, whereas sniffers are passive and can only observe a communication. Therefore proxies are suitable for assessment and testing purposes because they provide flexibilities to deal with the underlying communication of web applications.

5.5.1 WebScarab

WebScarab is a framework for analyzing applications that communicate using the HTTP and HTTPS protocols. It is written in Java, and is thus portable to many platforms. WebScarab has several modes of operation, implemented by a number of plugins. In its most common usage, WebScarab operates as an intercepting proxy, allowing the operator to review and modify requests

created by the browser before they are sent to the server, and to review and modify responses returned from the server before they are received by the browser. WebScarab is able to intercept both HTTP and HTTPS communication. The operator can also review the conversations (requests and responses) that have passed through WebScarab.[1]

5.5.2 Burpsuite

Burp suite allows an attacker to combine manual and automated techniques to enumerate, analyze, attack and exploit web applications. The various burp tools work together effectively to share information and allow findings identified within one tool to form the basis of an attack using another.

5.6 Network sniffer

In this section we aim to give an overview of the main Network sniffer tools. A network packet analyzer, also called sniffer, is a software that capture network packets and will display the result data as detailed as possible. As data streams flow across the network, the sniffer captures each packet and, if needed, decodes the packet's raw data, showing the values of various fields in the packet. Then, analyzes its content according to the appropriate RFC or other specifications. The captured informations are decoded from raw digital form into a human-readable format that permits users of the protocol analyzer to easily review the exchanged information.

5.6.1 Wireshark

Wireshark (official web page at: http://www.wireshark.org/) is a free open-source network packet analyzer, but before going into details of this software, we first want to mention that a network analyzer is not only a malicious sniffer that can be used for hacking only. It has several different aspects which can be used for different purpose. For example, Wireshark can be used to analyze network problem or network intrusion attempts. With this sniffer we can also identify and isolate exploited systems or simply monitoring our network for gather and report network statistics. This versatility is surely an added value for a software that we have used for scouting our case studies for better understanding how they can be modeled and how they work.

However, talking about major features of Wireshark, (originally called Ethereal) we need to write that it is cross-platform. Using the GTK+ widget

[1] description taken from https://www.owasp.org/index.php/Category:OWASP_WebScarab_Project.

toolkit to implement its user interface, and using pcap to capture packets, it runs on various Unix-like operating systems including Linux, Mac OS X, BSD, and Solaris, and on Microsoft Windows.

Talking about the features, it identifies automatically all ethernet and Wi-Fi interfaces of the system that runs it, and gives the possibility of starting a capture phase on all of them. So, without wasting time on configuring it you can easily start catching and analyzing the traffic. Furthermore, Wireshark automatically moves the network interface into the appropriate mode for catching packets: monitor mode or promiscuos mode. The promiscuous mode is a particular mode in which the NIC (Network Interface Controller) can pass *all* the captured packets to the CPU (Central Process Unit) and not just frames involving this machine as sender or receiver. The monitor mode, or RFMON (Radio Frequency MONitor) mode, do the same thing as the promiscuos do but while promiscuos is for wired NIC, the monitor mode is for wireless network.

On *Functionality*, this software is similar to tcpdump (offical web page at: www.tcpdump.org) but it has also a useful GUI (Graphical User Interface). This user friendly frontend helps every kind of users to understand which kind of packets are traveling in his net by the use of different colors and filters that can refine the search.

Since the SPaCIoS project focuses on security at provision and consumption level, we want to underline also how we can use this software for security testing. As mentioned before it is mainly used for the scouting phase of a software that is the preliminary step made for understanding how the software works. Wireshark can be used for both theoretical and practical security approach. In fact, if we want to create a model of a software, Wireshark can give us a picture of the structure of an architecture (distributed, Software oriented and so on). Otherwise, if we want to perform a practical penetration test, we need to do this scouting phase for (better) understanding informations like IP addresses, MAC addresses, number of participant or users and the like.

Another purpose can be performing timing attacks. A timing attack is a side channel attack in which the attacker attempts to compromise a cryptosystem by analyzing the time taken to execute cryptographic algorithms. One of the registered values is “Time” with which we can understand some interesting data in order to extend the knowledge needed for performing this kind of attack.

Since Wireshark has an easy to use graphical front-end, it does not require a configuration phase and has lots of interesting functionality and it is also a freeware open-source software, it can be highlighted as one of the best packet analyzer available today, and it is the reference one in Open Source

development.

5.7 Future Development Plan

There are two purposes in SPaCIoS industrial migration with respect to tools:

- (P1) usage of SPaCIoS tools for industrial applications;
- (P2) adoption of SPaCIoS tools into the industrial security validation activities.

While considering industrial migration of SPaCIoS tools, we should keep in mind the following facts. Although existing security validation tools are surveyed and compared, the purpose of SPaCIoS, especially of WP4, is not to develop a security validation tool that includes the features of all the other state-of-the-art validation tools. Our belief is that SPaCIoS can provide a complementary validation technology that is not featured by other existing tools, and thus SPaCIoS tools and existing security validation tools could be used together to achieve a better understanding of application and service security and thus better applications and services with respect to security.

In the following, we analyze how the two goals could be achieved in different migration modes aforementioned.

- In the consultancy mode, SPaCIoS tools are used by formal method experts to validate industrial applications and systems. It is quite straightforward that these activities contribute to (P1). But in order to achieve it, there are still some works to be done. First, test case lifecycle management needs to be supported. Test cases generated need to be stored in a library, to be adapted according to different deployment scenarios of the same applications or services, to be executed multi times during the development lifecycle of the applications or services, to be adjusted by the tester during execution to investigate certain aspects of SUT. Second, according to different runtime environments, proper test drivers need to be developed.
- In the domain-specific automated mode, SPaCIoS tools are used by industrial security analysts directly to validate applications and services. These activities can contribute to both (P1) and (P2), while this requires addressing the challenges aforementioned and a higher usability of the tool. We are not interested in re-developing a proxy or another vulnerability scanner. Ideally we would like to leverage on those tools that already have these features and exploit those features for SPaCIoS. This could result in, e.g., a SPaCIoS "plug-in" for WebScarab

or some tool else. This is a very interesting direction to investigate as for instance WebScarab could be already used in the industrial environment, so that adoption of SPaCIoS may be easier. We need to investigate more on whether this can be done. Perhaps it will be necessary to invest towards graphical editors for ASLan++ or USDL and debugging features to make easier the specification of the model, especially when we target (P2).

- The activities of the in-between mode can contribute to both (P1) and (P2). The directions we need to investigate are very similar to what was discussed for the domain-specific automated mode, but with the advantage that the modeling challenge is significantly mitigated.

6 Conclusions

In this deliverable, we report activities related to SPaCIoS industrial migration. According to three different migration mode foreseen, based on the results obtained from the questionnaire, we describe future development plans with respect to service description language, test case description language, and security validation tool.

In the next project year, we will work closely with business units to identify the most promising directions to investigate, and try to migrate SPaCIoS results obtained to industrial environments.

A Questionnaire

In this appendix, we give the complete questionnaire that we developed and distributed.

QUESTIONNAIRE

Modeling Security Aspects and Security Validation Tools for Web-based Applications

SPaCIoS: Secure Provision and Consumption in the Internet of Services

Project no. 257876, FP7-ICT-2009-5, ICT-2009.1.4: Trustworthy ICT
01/10/2010 – 30/09/2013

Website: http://www.spacios.eu/

Date: 2011-07-11

This questionnaire has been developed in the context of the SPaCIoS project and is intended to elicit user requirements from security analysts on the use of description techniques for security aspects and security validation tools for web-based applications.

These user requirements will be carefully considered in the SPaCIoS project in order to design and develop the SPaCIoS security testing platform to better support security analysts in their work.

Please answer the questions below, tick the appropriate boxes, and fill in the requested information. Filling in this questionnaire should take around 20 minutes.

Your feedback is essential to influence the development of a next generation of security testing tools. All individual answers we receive are treated strongly confidentially and not distributed to any third person. We will produce a summary of the findings, which will be made accessible to the SPaCIoS project partners exclusively (Università di Verona, ETH Zurich, Institut Polytechnique de Grenoble, Karlsruher Institut für Technologie, Università di Genova, SAP AG, Siemens AG).

Thank you for your contribution.

1 Technology of Web-based Applications

Which **implementation techniques** do you mainly use to build your web-based applications? (Please tick the appropriate boxes)

- Services:
  ☐ SOAP ☐ WSDL ☐ REST
- Application frameworks:
  ☐ J2EE/JEE ☐ ASP.NET ☐ PHP
- Dynamic web :
  ☐ AJAX ☐ JSON ☐ XML
- Others (Please specify):
  .....
  .....
  .....

2 Security Requirements

2.1 What kinds of **security goals** are mainly relevant in security testing of your web-based applications? For each category, please rate them accordingly to their relative importance in your projects and provide one or more examples, if applicable.

☐ **Authentication**
Authentication addresses the question: who are you? It is the process of uniquely identifying the clients of your applications and services. These might be end users, other services, processes, or computers. In security parlance, authenticated clients are referred to as principals.

(insignificant) – (light use) – (neutral) – (important) – (highly important)

Examples:
.....
.....
.....

☐ **Authorization**
Authorization addresses the question: what can you do? It is the process that governs the resources and operations that the authenticated client is permitted to access. Resources include files, databases, tables, rows, and so on, together with system-level resources such as registry keys and configuration data. Operations include performing transactions such as purchasing a product, transferring money from one account to another, or increasing a customer's credit rating.

(insignificant) – (light use) – (neutral) – (important) – (highly important)

Examples:
.....
.....
.....

☐ **Auditing**
Effective auditing and logging is the key to non-repudiation. Non-repudiation guarantees that a

user cannot deny performing an operation or initiating a transaction. For example, in an e-commerce system, non-repudiation mechanisms are required to make sure that a consumer cannot deny ordering 100 copies of a particular book.

(insignificant) – (light use) – (neutral) – (important) – (highly important)

Examples:
.....
.....
.....

☐ **Confidentiality**
Confidentiality is the process of making sure that data remains private and confidential, and that it cannot be viewed by unauthorized users or eavesdroppers who monitor the flow of traffic across a network. Encryption is frequently used to enforce confidentiality.

(insignificant) – (light use) – (neutral) – (important) – (highly important)

Examples:
.....
.....
.....

☐ **Integrity**
Integrity is the guarantee that data is protected from accidental or deliberate (malicious) modification. Integrity for data in transit is typically provided by using hashing techniques and message authentication codes.

(insignificant) – (light use) – (neutral) – (important) – (highly important)

Examples:
.....
.....
.....

☐ **Availability**
From a security perspective, availability means that systems remain available for legitimate users. The goal for many attackers with denial of service (DoS) attacks is to crash an application or to make sure that the application is sufficiently overwhelmed so that other users cannot access it.

(insignificant) – (light use) – (neutral) – (important) – (highly important)

Examples:
.....
.....
.....

☐ **Further security goals** (Please specify)
.....
.....
.....

(insignificant) – (light use) – (neutral) – (important) – (highly important)

2.2 What kinds of **security vulnerabilities** are mainly relevant in security testing of your web-based applications? (Please tick boxes)

☐ Cross Site Scripting (XSS) (reflected and stored)

☐ Cross Site Request Forgery (CSRF)

☐ Injection (e.g., SQL injection, command injection)

☐ Session Management and Session Fixation

☐ Insecure Direct Object References

☐ Security Misconfiguration

☐ Insecure Cryptographic Storage

☐ Failure to Restrict URL Access

☐ Insufficient Transport Layer Protection

☐ Unvalidated Redirects and Forwards

☐ Others (Please specify)

.....
.....
.....

2.3 **Security validation process**
Do you have an established process to validate security properties? (yes) / (no)

If yes, in which phase of the development process the validation is done?

(req. analysis) (design) (coding) (integration) (system testing) (deployment)

Would you invest in establishing a security validation process or improve an existing one?

(no) / (maybe in future) / (yes in future) / (definitely now)

3 Descriptions of Security Requirements

What techniques do you use to describe your security requirements? Please explain when necessary and provide a couple of examples.

☐ **Security goals** as mentioned in Section 2.1
Descriptions are
☐ implicitly defined / ☐ part of the general requirement document /
☐ written as a dedicated security specification.
.....
.....
.....

☐ **Security vulnerabilities** as mentioned in Section 2.2
Descriptions are
☐ implicitly defined / ☐ part of the general requirement document /
☐ written as a dedicated security specification.
.....
.....
.....

4 Security Validation Tools

Which tools do you use to validate the security requirements stated above?

☐ **Source code analyzer**, e.g. Fortify, Parasoft JTest
.....
.....
.....
.....

- In which phase of your system development process do you use them?

  (req. analysis) (design) (coding) (integration) (system testing) (deployment)

- How critical are the tools in your security testing process?

  (unimportant) – (minor use) – (neutral) – (important) – (mandatory)

FP7-ICT-2009-5, ICT-2009.1.4
Project No. 257876

☐ **Web application vulnerability scanner**, e.g. Appscan, WebInspect

.....
.....
.....
.....

- In which phase of your system development process do you use them?

  (req. analysis) (design) (coding) (integration) (system testing) (deployment)

- How critical are the tools in your security testing process?
  (unimportant) – (minor use) – (neutral) – (important) – (mandatory)

☐ **Proxy tools**, e.g. OWASP WebScarab

.....
.....
.....
.....

- In which phase of your system development process do you use them?

  (req. analysis) (design) (coding) (integration) (system testing) (deployment)

- How critical are the tools in your security testing process?
  (unimportant) – (minor use) – (neutral) – (important) – (mandatory)

☐ **Others** (e.g. fuzzing tools, please explain)

.....
.....
.....
.....

- In which phase of your system development process do you use them?

  (req. analysis) (design) (coding) (integration) (system testing) (deployment)

- How critical are the tools in your security testing process?
  (unimportant) – (minor use) – (neutral) – (important) – (mandatory)

5 Open Issues

Please describe open issues that you face in your daily work with respect to the description of security aspects and the validation of these aspects.

Issues in the description of security goals or vulnerabilities.
In which stage/activity of your development process, support by appropriate tools needs to be introduced or improved with respect to security requirement identification, security requirement description, security requirement concretization, etc.?

.....
.....
.....
.....
.....
.....
.....
.....

Issues in the validation of security goals or vulnerabilities.
In which phase of your development process support by appropriate security validation tools needs to be introduced or improved?

.....
.....
.....
.....
.....
.....
.....
.....

Issues related to additional requirements imposed by a certification authority.
For example, compliance with the Common Criteria for Information Technology Security Evaluation:

.....
.....
.....
.....
.....
.....
.....
.....

References

[1] OWASP Testing Guide: Black Box testing DOM-based Cross site scripting. https://www.owasp.org/index.php/Testing_for_DOM-based_Cross_site_scripting_%28OWASP-DV-003%29#Black_Box_testing_and_example.

[2] Web Application Vulnerability Scanner Evaluation Project (wavsep). http://code.google.com/p/wavsep/.

[3] Web Vulnerability Scanners Evaluation. http://anantasec.blogspot.com/, January 2009.

[4] AVANTSSAR. Deliverable 2.3 (update): ASLan++ specification and tutorial, 2011. Available at http://www.avantssar.eu.

[5] Shay Chen. The Scanning Legion: Web Application Scanners Accuracy Assessment & Feature Comparison . http://sectooladdict.blogspot.com/2011/08/commercial-web-application-scanner.html, August 2011.

[6] Internet of Services. USDL Specifications. http://www.internet-of-services.com/index.php?id=570&L=0, 2011.

[7] SAP. The Extended Computer Aided Test Tool. http://www.sdn.sap.com/irj/sdn/ecatt, 2011.

[8] ESTI Standard. *Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language*. ESTI, 2010. Available at http://www.ttcn-3.org/.

[9] The World Wide Web Consortium. W3C Unified Service Description Language Incubator Group. http://www.w3.org/2005/Incubator/usdl/, 2011.