The 2011 CESSA project retrospective by SAP Research

Applies to:

SAP Netweaver, SAP Netweaver Neo

Summary

In the CESSA project SAP is providing innovative solutions to help developers to escape from web security pitfalls. This white paper presents the latest results we achieved in and the remaining challenges we will tackle in providing web services and web applications security.

Author(s): Jean-Christophe Pazzaglia, Anderson Santana de Oliveira, Theodoor Scholte, Gabriel Serme, Jakub Sendor, Julien Massiera

Company: SAP Research

Created on: 10 January 2012

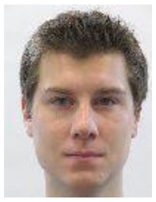
Author Bios

Dr. Jean-Christophe Pazzaglia. Since January 2010, Jean-Christophe is the manager of the security engineering team at SAP research S&T practice. Jean-Christophe worked on international environments since 1997 and live in 4 European countries France, UK, Italy and Cyprus where he held several positions between research labs (Institut Eurécom, CRS4, Cardiff University, I3S) and private companies (Albourne Partners, Mediatech and Softeam). Jean-Christophe joined eventually SAP Research in 2006 to take care of all development activities related to the Security and Trust Research Program. In this respect, he is involved in various European founded Initiatives (Serenity, PrimeLife, TAS 3,...) together with internal activities related to the Security aspects of Software development. Jean-Christophe holds a Ph.D. in Computer Science from the University of Nice-Sophia Antipolis, France (1997) and graduates the Essentials of Management program of University of St Gallen (2009).

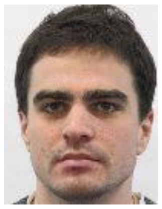
Anderson Santana de Oliveira is a researcher and team leader for CESSA and was responsible for the technical coordination of the FP7 MASTER project. During his Ph.D. studies at INRIA, he participated to the NoE REWERSE and subsequently taught formal methods, automata theory and compilers at the UFERSA university in Brazil. His research are concentrated on the application of formal reasoning to security policies and he has been a reviewer for journals and conferences on formal methods and security.

Theodoor Scholte is a researcher at SAP Research S&T practice. He is a member of the ANR CESSA project and member of the EU FP7 PoSecCo project. He has participated to the EU FP7 MASTER project, where he worked on the definition and development of a monitoring framework for SOA. His research interests are in the domains of security and software engineering. Currently, he is a PhD student at Institute Eurecom and he is going to defend his PhD thesis in Q2 2012.

Gabriel Serme is a research associate in SAP Research Sophia-Antipolis and a PhD candidate in Eurecom since December 2009. He is member of the ANR CESSA Project and is involved in activities to support systematic description and application of security properties throughout the software development lifecycle. For this purpose, he is investigating on the AOSD and AOP paradigm to create a better integration of cross-cutting concerns. Gabriel holds both master degree from the university of Nice-Sophia Antipolis and engineering diploma from Polytech'Nice-Sophia in Computer Science.



Jakub Sendor is a developer working in SAP Research since April 2011. In addition to the work on the transfer of the policy engine prototype to the Business Web, he is also involved in the secure programming trainings for Java as a member of Security Task Force team. He has previously worked in the development of the privacy policy engine for the EU Primelife project during his six-month internship that concluded his master degree studies at Eurecom/Telecom ParisTech and AGH University of Science and Technology in Cracow, Poland.



Julien Massiera is a junior development engineer at SAP Research since September 2011. After his internship in the CESSA project, he conducted the installation and documentation quality testing in the EU TAS3 security architecture components. He holds a MIAGE master degree and two bachelor degrees in computer science from the University of Nice Sophia-Antipolis where he learned about networks and computer science.

Table of Contents

The CESSA Project

Service oriented architectures are consolidated as a paradigm for software development since some years now. With the shift to the cloud, service oriented software demand for responsive approaches to cope with the frequently changing security requirements and compliance constraints.

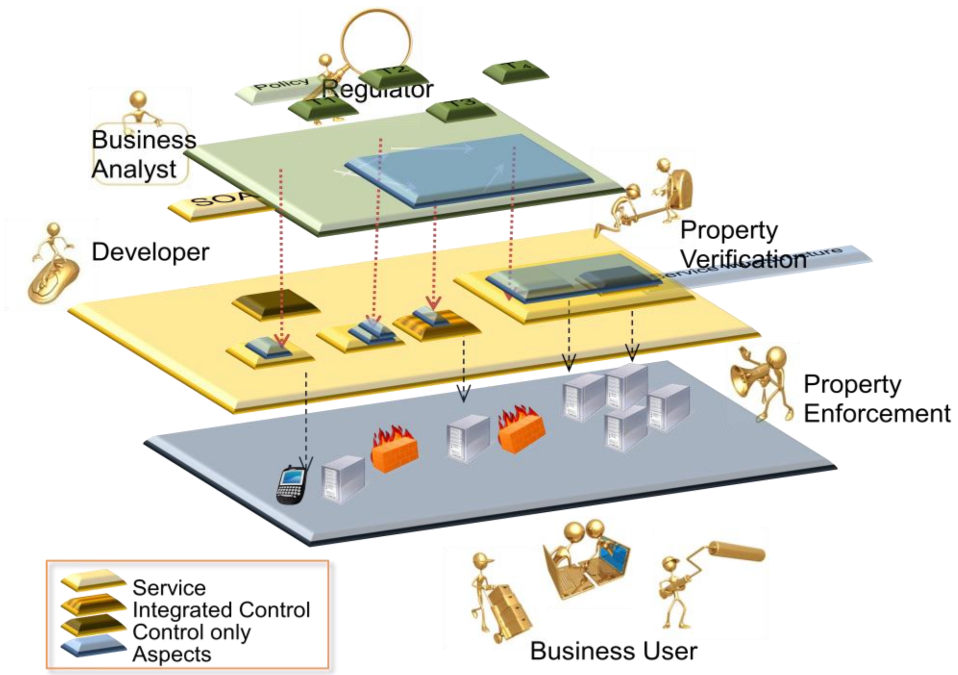


Figure 1 - Managing compliance and security requirements - the big picture

Figure 1 Illustrates the objectives we want to achieve in CESSA. At the upper level, business analysts specify security policies according to the recommendations of different regulators. Then developers use aspect-oriented software development techniques to implement these policies across different administrative and technological domains. In parallel, some certification procedures can be undertaken. At the concrete level, the security functionalities are synthesized, not only at the server side, with for instance computer clusters, but also at the client side, with for instance mobile devices. Finally, the business users can benefit from the services while trusting the whole architecture, which by construction enforces the different security policies.

Key figures

The essential information about the project and the consortium are given in the table below. Please refer to the CESSA website for updated information and to follow up on its results.

Start date	Duration	Total budget	Funding agency
January 2010	36 months	2.1 M€	French National Research Agency (ANR)

Partners

- SAP
- École de Mines de Nantes
- Institut Eurecom
- IS2T

Our Contributions to CESSA In 2011

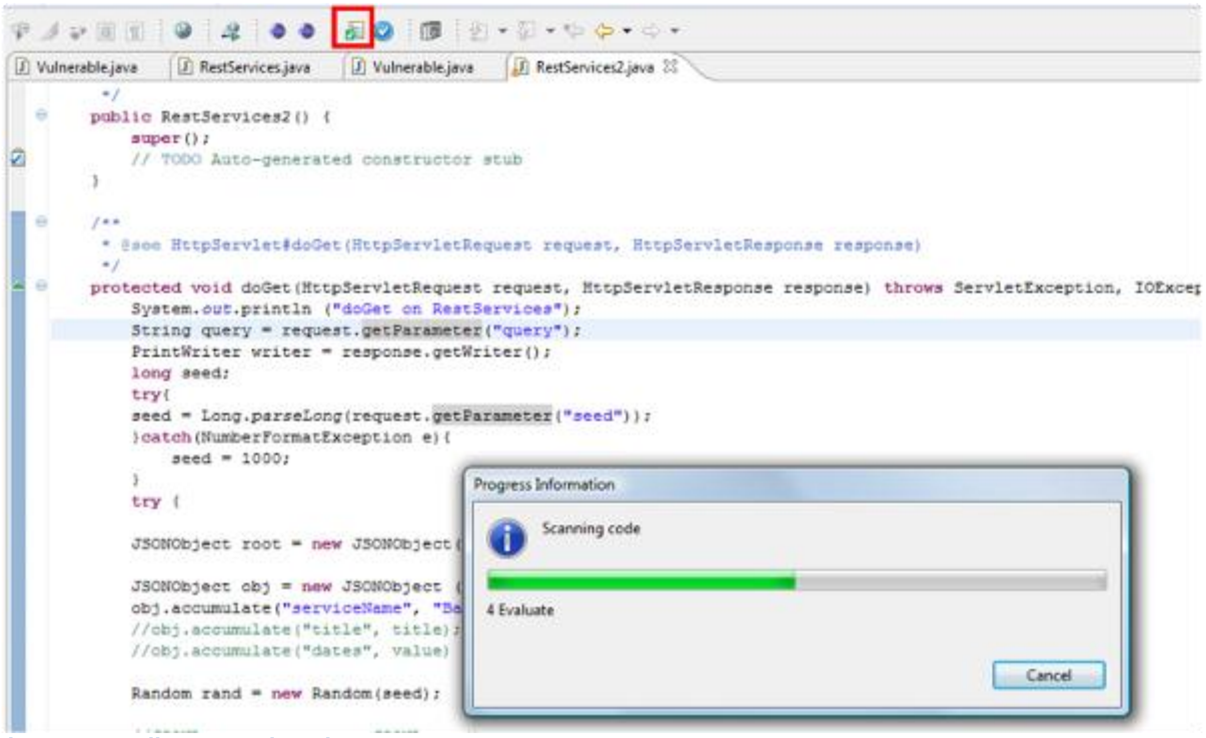
We can summarize in one sentence the work we are doing in CESSA: “How to assist developers, to identify the security concerns and to automate as much as possible secure development practices for service-oriented software. In the sections below we highlight some of our most relevant developments.

Security Standards Automation

After a decade of existence, still, Cross-Site Scripting(XSS), SQL Injection and other of types of security vulnerabilities associated to input validation can cause severe damages when exploited. To analyze this fact, Scholte et al [4] conducted an empirical study that shows that the number of reported vulnerabilities is not decreasing. Taking these results into consideration, three deficiencies can be highlighted: a lack in upskilling developers to write secure code, a high ratio of false positive findings in security code scanners and an erroneous planning for security corrections.

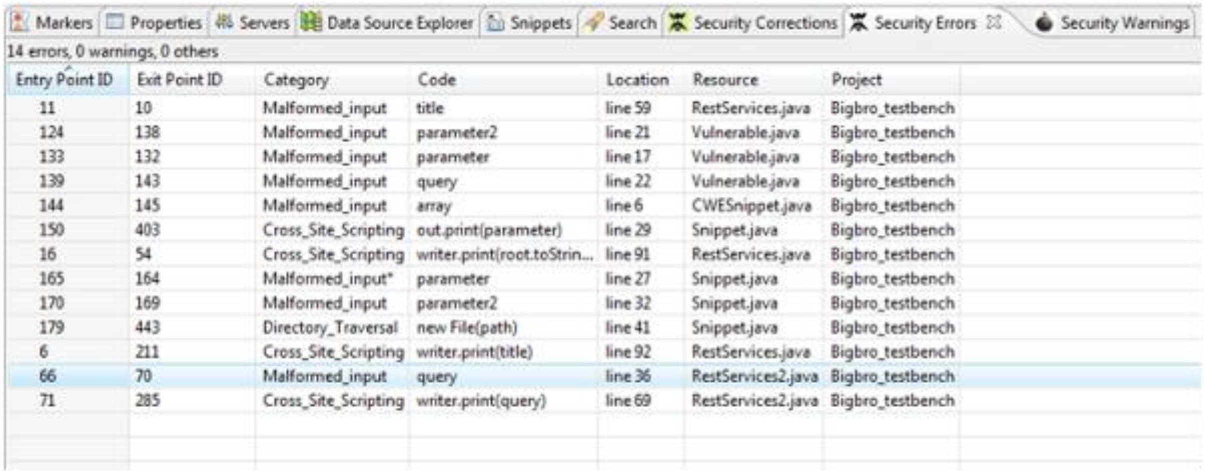
Gabriel Serme, Paul El'Khoury and Marco Guarnieri, a former intern at SAP Labs France, showed how to overcome these deficiencies using the Eclipse platform and the JDT compiler in a proper tooling. This work was published in [1]. They have written a static analyzer that assists developers to report these security vulnerabilities. Secondly they introduced an innovative approach by integrating an Aspect Oriented tool for semi-automated correction of these findings. Both tools are designed within an architecture that is monitored by security experts and particularly adequate for agile development.

Snapshots of the tool are shown in the pictures Figure 2 and Figure 3, which gives direct access to detected flaws and global overview on system vulnerabilities. For more information, follow Gabriel’s posts in his SDN blog.



```
     */
    public RestServices2() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOExcep
        System.out.println ("doGet on RestServices");
        String query = request.getParameter("query");
        PrintWriter writer = response.getWriter();
        long seed;
        try{
        seed = Long.parseLong(request.getParameter("seed"));
        }catch(NumberFormatException e){
            seed = 1000;
        }
        try {

        JSONObject root = new JSONObject(

        JSONObject obj = new JSONObject (
        obj.accumulate("serviceName", "Be
        //obj.accumulate("title", title);
        //obj.accumulate("dates", value)

        Random rand = new Random(seed);
```

Figure 2 - Coding Scanning view



14 errors, 0 warnings, 0 others

Entry Point ID	Exit Point ID	Category	Code	Location	Resource	Project
11	10	Malformed_input	title	line 59	RestServices.java	Bigbro_testbench
124	138	Malformed_input	parameter2	line 21	Vulnerable.java	Bigbro_testbench
133	132	Malformed_input	parameter	line 17	Vulnerable.java	Bigbro_testbench
139	143	Malformed_input	query	line 22	Vulnerable.java	Bigbro_testbench
144	145	Malformed_input	array	line 6	CWESnippet.java	Bigbro_testbench
150	403	Cross_Site_Scripting	out.print(parameter)	line 29	Snippet.java	Bigbro_testbench
16	54	Cross_Site_Scripting	writer.print(root.toStrin...	line 91	RestServices.java	Bigbro_testbench
165	164	Malformed_input*	parameter	line 27	Snippet.java	Bigbro_testbench
170	169	Malformed_input	parameter2	line 32	Snippet.java	Bigbro_testbench
179	443	Directory_Traversal	new File(path)	line 41	Snippet.java	Bigbro_testbench
6	211	Cross_Site_Scripting	writer.print(title)	line 92	RestServices.java	Bigbro_testbench
66	70	Malformed_input	query	line 36	RestServices2.java	Bigbro_testbench
71	285	Cross_Site_Scripting	writer.print(query)	line 69	RestServices2.java	Bigbro_testbench

Figure 3 - Scanning results as seen by developers

RESTful web service security

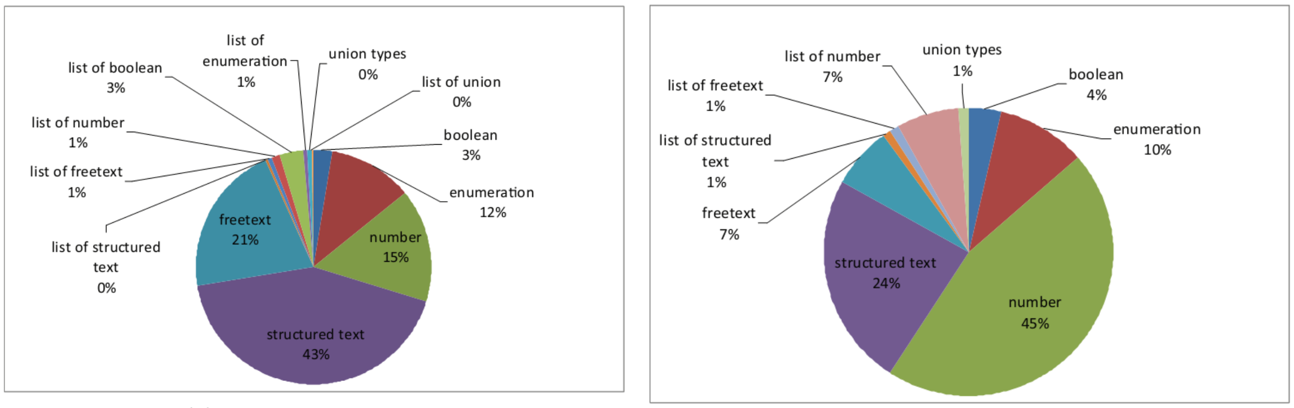
Security for Resource Oriented Services is far from reaching a mature stage. A growing number of significative REST (REpresentational State Transfer) API's were recently released such as the Twitter API, the Google Search API, and the SAP Streamwork's integration API, among others. Each of these implement their own security mechanisms, often in a “homemade” manner, potentially containing security bugs. It is hard for the developers of such systems to design and deploy well adapted security mechanisms, as there is currently no industrial standard for RESTful web services. Inspired from AOP we developed a cutting edge framework to facilitate the design of secure services relying on pure REST principles. It relies in intercepting he workflow in the application server in order to apply security countermeasures in the message exchange. We will release more information about this work very soon.

Understanding injection attacks in web services and applications

Web services and applications have become essential in our daily lives, and millions of users access these applications to communicate, socialize, and perform financial transactions. Unfortunately, due to their increased popularity as well as the high value data they expose, web applications have also become common targets for attackers.

In order to gain deeper insights into the reasons behind common vulnerabilities, Scholte et al. [5] analyzed around 3,933 cross-site scripting (XSS) and 3,758 SQL injection vulnerabilities affecting applications written in popular languages such as PHP, Python, ASP, and Java. For more than 800 of these vulnerabilities, they manually extracted and analyzed the code responsible for handling the input, and determined the type of the a selected parameter (e.g., boolean, integer, or string). Furthermore, we studied 79 web application frameworks available for many popular programming languages. Our results suggest that most SQL injection and a significant number of XSS vulnerabilities can be prevented using straightforward validation mechanisms based on common data types.

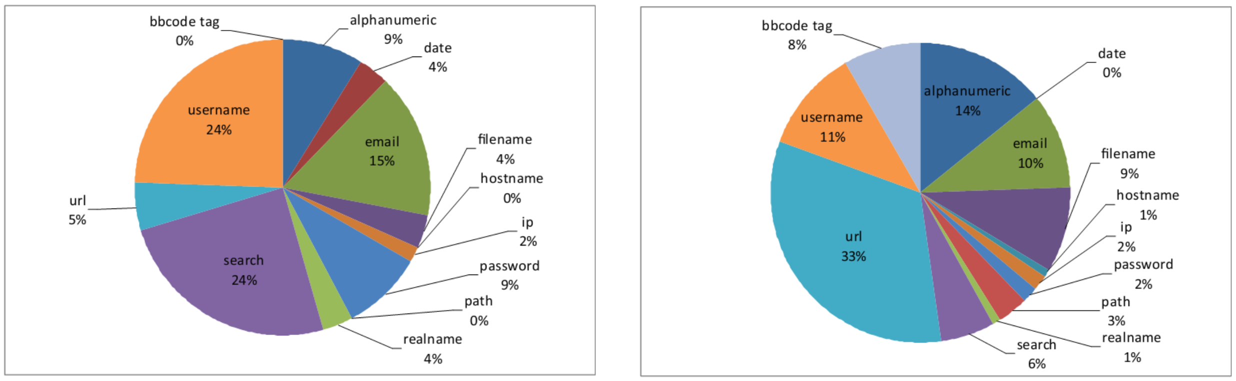
In [2] more than 3500 XSS and 3500 SQL injection vulnerability reports were analyzed in detail. The authors have also studied the validation functions provided by 79 web application frameworks. To our knowledge, this is **the largest empirical study of its kind conducted to date**. Our study suggests that many SQL injection and XSS vulnerabilities can be prevented if web languages and frameworks would support the enforcement of common data types such as integer, boolean, and specific types of strings such as an e-mail or a delivery address. Figure 4 (a) and (b) shows an overview of the types corresponding to input parameters vulnerable to XSS and SQL injection. Most of the vulnerable parameters had one of the following types: boolean, numeric, structured text, free text, enumeration, or union.



(a) XSS vulnerabilities. (b) SQL injection vulnerabilities.

Figure 4 - Data types corresponding to vulnerable input parameters

Figure 5 b and a show a detailed overview of which particular “structured text" is responsible for most of the XSS and SQL injection vulnerabilities. The graph shows that web applications would benefit from input validation routines that are able to sanitize complex data such as URLs, usernames, and email addresses, since these data classes are often used as attack vectors



(a) XSS vulnerabilities. (b) SQL injection vulnerabilities.

Figure 5 - Structured string corresponding to vulnerable in-put parameters.

The work findings suggest that many SQL injection and XSS could easily be prevented if web languages and frameworks would be able to automatically enforce common data types such as integer, boolean, and specific types of strings such as e-mails and URLs.

Privacy enforcement in Netweaver Neo

We have designed a framework able to modify, at deployment time, the architectural elements (as databases) enriching it with the elements in order to enforce user privacy preferences, such that any new application deployed on the modified platform would benefit from private-aware data handling. The approach, published in [3], is depicted in Figure 6.

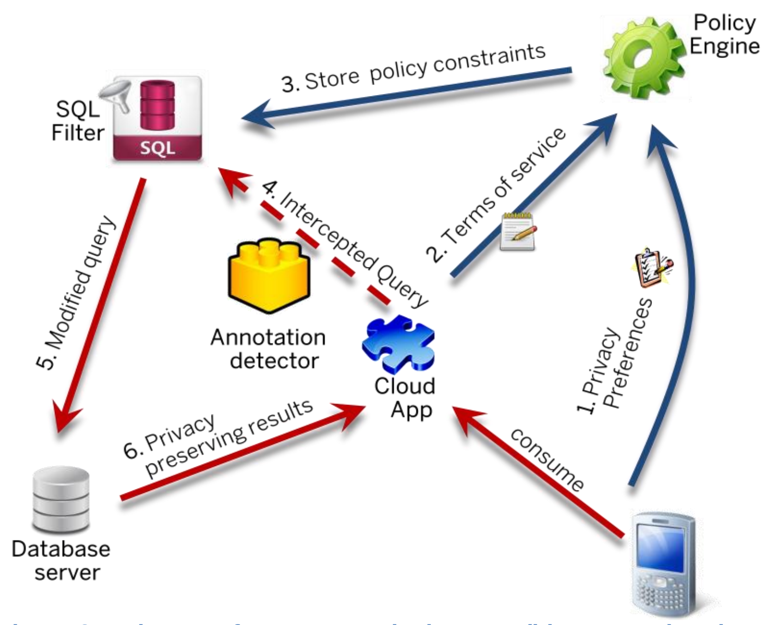


Figure 6 - Privacy enforcement at deployment (blue arrows) and run-time (red arrows)

Usually, the application code handling privacy related data is scattered and tangled over the application, being difficult to handle. The operations performed on this data are typically what we call a cross-cutting concern in aspect-oriented programming (AOP).

We provide a simplified process for the application developer to rapidly achieve data protection compliance. It consists in adding meta-information to the application code via an annotation mechanism. The annotations include type of data to protect and the purpose of the data usage.

On their turn, cloud application developers would simply annotate the application code to indicate where Personally Identifiable Information (PII) is being treated along with the purpose of the corresponding operation, instead of making explicit API calls to handle private data.

The modifications we bring are transparent from an application point-of-view as components propose the same set of APIs than in traditional platforms. It gives an added-value to respect legacy-compatibility while allowing privacy management when needed.

An example of annotation used by applications is depicted in Figure 7. Developers are required to add annotation to objects, such as @PII annotation at the definition of the Java classes. This annotation indicates that the class comprises one or more field having privacy-constraints. There are for example limitations for several purposes applied to the shopping list.

```
@Entity
@PII
public class ShoppingHistory implements Serializable {  … …  }
@Info(purpose="marketing", recipient="Store")
public List<T> findObjects(Class<T> className) {  … …  }
```

Figure 7 Annotating persistent data as PII

We are experimenting with the Netweaver Neo platform. This work was performed by Peng Yu, intern at SAP Labs France, Jakub Sendor, Gabriel Serme and Anderson Santana de Oliveira.

The road ahead: research challenges

Several topics will deserve more research and development during 2012:

- Enforcement of privacy obligations in the business web
- Performing inference on the input parameter types for web services and applications
- Developing a language to orchestrate distributed aspects for services, and more generally, software components
- Creation of a RESTful web service security toolkit integrated to Eclipse leveraging security in Netweaver Neo

Moreover, we will also address open research questions, identified in [4], in particular in the formal modeling of security properties in SOA.

References

1. Marco Guarnieri, Paul El-Khoury and Gabriel Serme. ***Security Vulnerabilities Detection and Protection Using Eclipse***. Proceedings of ECLIPSE-IT 2011, (2011)
2. Muhammad Sabir Idrees, Gabriel Serme, Yves Roudier, Anderson Santana De Oliveira, Herve Grall and Mario Sudholt. ***Evolving Security Requirements in Multi-Layered Service-Oriented-Architectures***. 4th International Workshop on Autonomous and Spontaneous Security, in conjunction with the 16th annual European research event in Computer Security (ESORICS 2011) symposium, September 15-16, 2011, Leuven, Belgium, 09 2011.
3. Anderson Santana de Oliveira, Jakub Sendor, Gabriel Serme, and Peng Yu. ***Privacy Policy Enforcement for Cloud Platforms***, Poster, 3rd International Conference on Cloud Computing Technology and Science, IEEE (2011)
4. Theodoor Scholte, Davide Balzarotti, and Engin Kirda. ***Quo vadis? a study of the evolution of input validation vulnerabilities in web applications***. In Proceedings of Financial Cryptography and Data Security 2011, Lecture Notes in Computer Science, February 2011.
5. Theodoor Scholte, Davide Balzarotti, William Robertson, Engin Kirda. ***An Empirical Analysis of Input Validation Mechanisms in Web Applications and Languages***. Accepted at the ACM Symposium in Applied Computing. To Appear, 2012.

Copyright