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 D6.2.2
Migration to SAP and Siemens Business Units (Lessons Learned and Best Practices)

Abstract

This deliverable contains a summary of industrial migration experiences of SPaCIoS technologies, and retrospection of lessons learned and best practices recommended. test

Deliverable details

Deliverable version: *v1.0* Classification: *public*
Date of delivery: **this is a draft, 31.03.13** Due on: *30.09.2013*
Editors: *SAP and Siemens principal editors; UNIVR, ETH Zurich, TUM, INP and UNIGE secondary editors* Total pages: *28*

Project details

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

SEVENTH FRAMEWORK PROGRAMME

Contents

List of Figures

List of Tables

1 Introduction

This deliverable reports about our performed activities towards the transfer of the SPaCIoS results to our industrial partners SAP and Siemens.

During the first year of the project, we studied state-of-the-art service description languages and validation tools as well as we redacted a specific questionnaire to elicit user requirements from security analysts in industry (see SPaCIoS Deliverable D.6.2.1 [5] for more details).

The input collected from business units at SAP and Siemens via the questionnaire provided a clear picture of what industry uses and needs in terms of description techniques for security aspects, critical security properties and security validation tools for web-based applications. These user requirements have been 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.

Contextually, a number of migration activities have been selected and initiated in a strict collaboration with the business units at SAP and Siemens. While these activities are still on-going, short-term results are already emerging and, even more important, their execution is providing a lot of promising ideas and directions for a concrete exploitation of SPaCIoS in industry on the medium and long term despite the intrinsic challenges we already identified in Deliverable D.6.2.1 [5] (e.g., model specifications, performances).

All in all, the strategy, as we defined it in the first year of the project, is still valid and it is executed in the context of these transferring activities.

The deliverable starts with restating our industry migration strategy in Section 2. The activities under-going at SAP and Siemens are then discussed in Section 3.1 and Section 3.2 respectively. We conclude in Section 5 after some final remarks and lesson-learned so far in Section 4.

2 Industrial Migration Strategy

In this section, we restate what we discussed during the first review about the industrial migration strategy. The purpose is to map the activities performed in the second project year to the strategy.

The goal of the workpackage 6.2 is to expedite the transfer of SPaCIoS results to industry, including standardization organizations and open source communities. 80% of the effort is for migration to SAP and Siemens business units, while 20% is for migration to industrial interest groups and open-source communities. When defining the industrial migration strategy, we consider results from the questionnaire, as well as intrinsic technical challenges underlying the SPaCIoS approach. In a summary, our migration strategy has the following aspects:

- We think that addressing all the challenges for any industrial domain is unlikely.
- We foster opportunities for adoption of SPaCIoS results in industry by:
  - Use of the widely-used Eclipse development environment to run the SPaCIoS machinery so to, e.g., lower at minimum usability entry-barriers for developers;
  - Use of established standardized languages for testing (e.g. TTCN-3), to support test life-cycle management, and to support debugging of tests for easier fault identification;
  - Mixture of different kinds of migration activities including project-specific consulting, domain-specific tools, and in-between activities.

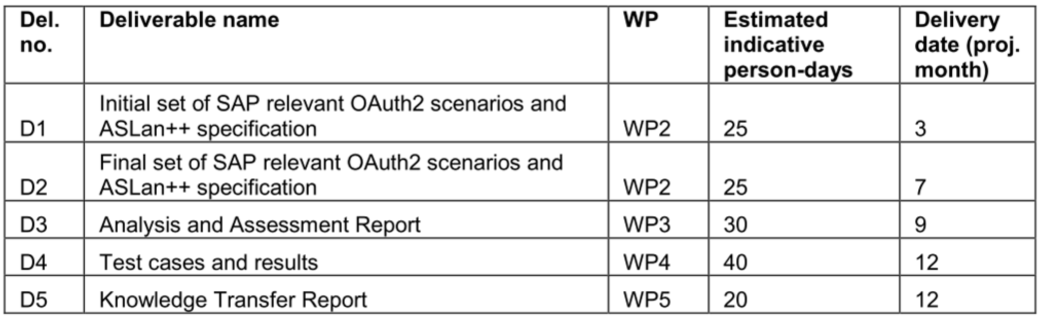
In the “project-specific consulting” activities, experts of the SPaCIoS approach act as formal method experts authoring the model, run the validation using the SPaCIoS tool, and report the results obtained. This kind of activities will be performed within the SPaCIoS timeframe.

In the “domain-specific tools” activities, we identify industrial domains where automated approach can be viable. One prerequisite of this kind of activities is that security models can be (partially) generated automatically, from other representations or models, or by model inference. This kind of activities may be performed beyond the SPaCIoS timeframe, while within the SPaCIoS timeframe, effort will be spent to pave the way.

In the “in-between” activities, some steps are performed by people from business units. The experts of the SPaCIoS approach provide formal security models (e.g., in ASLan++), or generated test suite, and the validation is performed by people from business units. This kind of activities will be performed within the SPaCIoS timeframe.

3 Industrial Migration Activities

In this section, we report on the list of the activities we have initiated, and map these activities to the strategy. While these activities are mainly run within the premises of our project industrial partners SAP and SIEMENS, some of their outcomes already reached the outside world communities of



Del. no.	Deliverable name	WP	Estimated indicative person-days	Delivery date (proj. month)
D1	Initial set of SAP relevant OAuth2 scenarios and ASLan++ specification	WP2	25	3
D2	Final set of SAP relevant OAuth2 scenarios and ASLan++ specification	WP2	25	7
D3	Analysis and Assessment Report	WP3	30	9
D4	Test cases and results	WP4	40	12
D5	Knowledge Transfer Report	WP5	20	12

Figure 1: SAP OAuth2 - Deliverables

standardization and open-source and we are optimistic other results can follow the same path. For instance, the SAML Version 2.0 Errata 05 was approved by OASIS on the 1st of May 2012, [3]. Section "E90: RelayState sanitization" modifies the specifications "SAML Bindings" [1] and "SAML Profile" [2] in several ways. The errata acknowledges the EU Projects AVANTSSAR, SPaCIoS, and SIAM for the identification of the problem, and assistance in drafting the material.

3.1 SAP's activities

The following SPaCIoS industrial migration activities have been carried out at SAP.

Security Validation of SAP OAuth2 This is the migration activity on which SAP invested more overall. SAP is developing its own implementation of OAuth2 that will be deployed within various SAP products e.g., the SAP NetWeaver Application Server. Based on its business context, SAP OAuth2 scenarios are quite different than both the standard printing service scenarios described in the OAuth2 specification and the scenarios implemented within Facebook, which is analyzed in WP5. One example is a variant of the OAuth 2-legged scenario in which a Security Token Service (STS) is added to issue the SAML Bearer Assertion for the OAuth client acting on behalf of a user. In this migration activity, we perform formal modeling, security validation, and testing of SAP OAuth2 using the various techniques developed in SPaCIoS project. In the year 2012, around 8 PMs have been spent for this activity. It has its own project structure containing 5 work packages agreed between SAP Research and SAP TIP Core SIM, which is the receiving business unit at SAP. Figures 1 and 2 presents the list of deliverables and the corresponding Gantt chart. Until now, more than 100 ASLan++ specifications have been

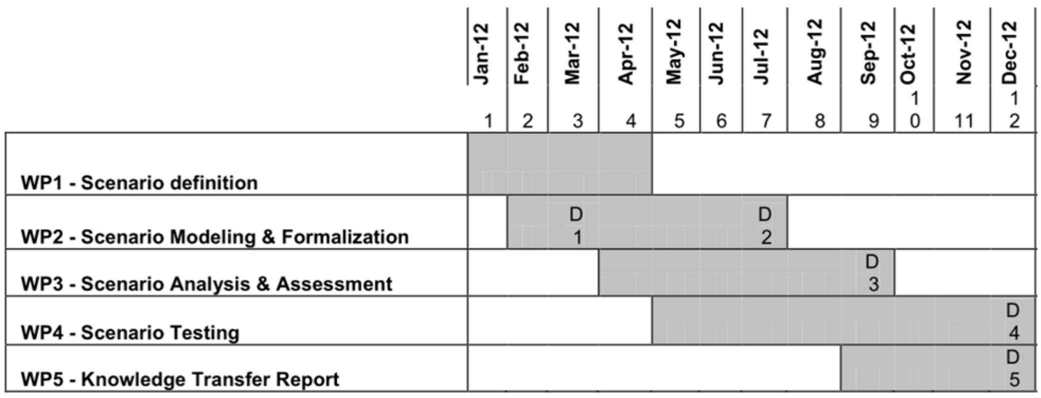


Figure 2: SAP OAuth2 - Gantt chart

built and analyzed. The Testing activity (WP4) started later than expected and various challenges emerged that require non-negligible effort to be solved.

For instance, test-case generation and execution for the SAP OAuth2 solution needs the tester to be able to simulate server-side activities such as message signature, message signature validation, assertion issuing, etc. While the Instrumentation-Based Testing Approach is general enough to conceptually support these situations, the corresponding proof-of-concept implementation required significant extensions.

Despite these challenges, various test cases have been already generated and executed, considering not only abstract traces representing potential security property violations, but also some capturing expected functional behaviors of the 2-legged protocol scenario.

Additional scenarios are in the radar-screen for this migration activity as well as the enhancement of the SPaCIoS tool to make it more usable for industry and/or usage of other SPaCIoS components to automatically generate and execute test-cases on the SAP OAuth2 scenarios. In this respect, we have extended this work toward 2013.

This migration activity belongs to the “project-specific consulting” migration mode. The modus-operandi is summarized in Figure 3. The SAP TIP Core SIM business unit provides us with the document specifications (including Message Sequence Chart (MSC), internal requirements and design decisions) of the SAP OAuth2 solution under-development, the detail on the business scenarios, etc. We process all these inputs and we create some formal specifications in ASLan++ that we validate against a few functional properties to be sure we really capture in the specifications what we have in mind. These specifications are then discussed with the business unit together with the various questions that could have emerged. We refine the specifications and we formalize the security properties corresponding to each one of the security requirements implicitly defined in the considered SAP OAuth2 business scenarios. We also produce variants of the formal specifications each

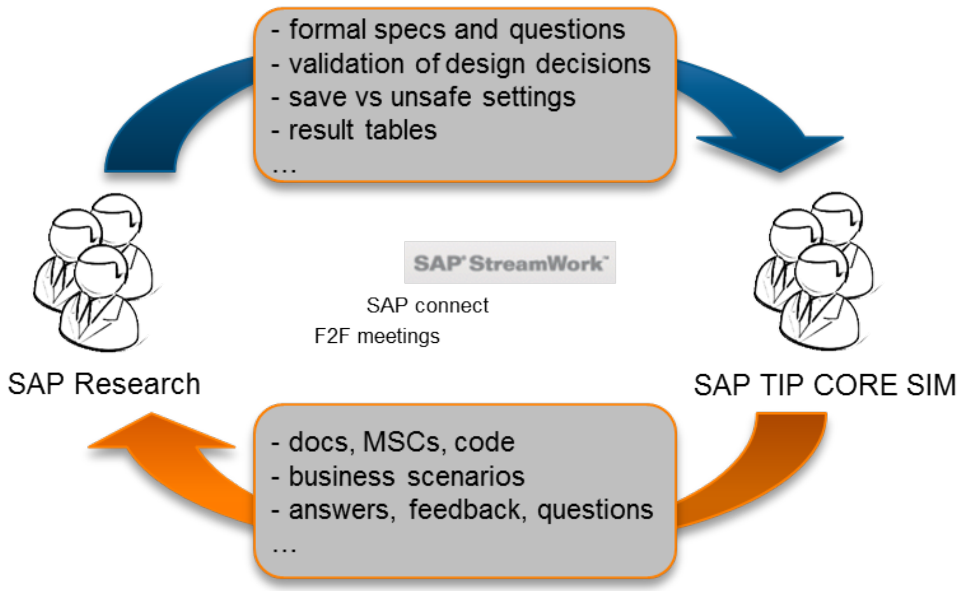
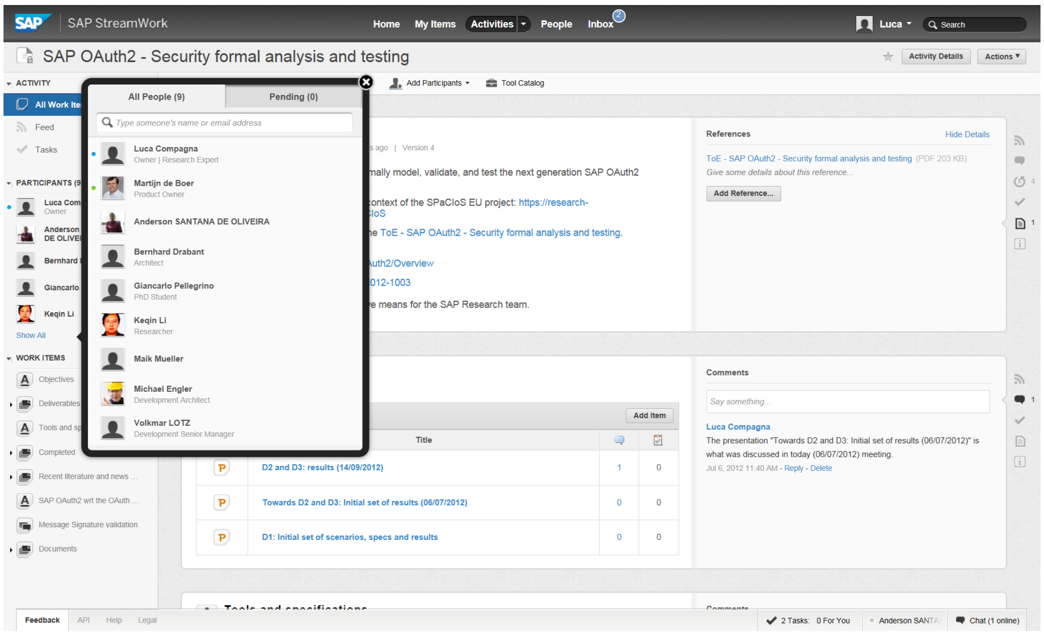


Figure 3: SAP OAuth2 - Collaboration mode

one capturing a potential setting related to the SAP OAuth2 scenarios. For instance, one specification could capture the situation in which the third party Security Token Service fails in authenticating the OAuth2 client so to evaluate the consequences of this setting on the overall scenario. Results obtained are regularly reported to the business unit and discussions take place to sort out emerging questions as well as interesting results and feedback. Operationally we use the SAP StreamWork as collaborative platform and SAP Connect to handle regular virtual meetings. SAP StreamWork is an on-demand enterprise collaboration tool from SAP AG released in March 2010. SAP StreamWork allows real-time collaboration focusing on business activities such as analyzing data, planning meetings, and making decisions. We created a specific SAP StreamWork activity for our internal migration (see Figure 4) to which we invited all the important stakeholders (no other people can access our activity). We mainly use SAP StreamWork to start virtual discussions (more effective than emails), to raise questions and take decisions, assign tasks, to keep track of the results and of the status. Email notifications allows each one of the team members to be immediately informed about the latest news.



SAP OAuth2 - Security formal analysis and testing

Luca Compagna
Owner | Research Expert

Martijn de Boer
Product Owner

Anderson SANTANA DE OLIVEIRA

Bernhard Drabant
Architect

Giancarlo Pellegrino
PhD Student

Keqin Li
Researcher

Maik Mueller

Michael Engler
Development Architect

Volkmar LOTZ
Development Senior Manager

s ago | Version 4

mally model, validate, and test the next generation SAP OAuth2

context of the SPaCIoS EU project: https://research-
CIoS

he ToE - SAP OAuth2 - Security formal analysis and testing.

uth2/Overview

012-1003

e means for the SAP Research team.

References — Hide Details

ToE - SAP OAuth2 - Security formal analysis and testing (PDF 203 KB)
Give some details about this reference...

	Title		
P	D2 and D3: results (14/09/2012)	1	0
P	Towards D2 and D3: Initial set of results (06/07/2012)	0	0
P	D1: Initial set of scenarios, specs and results	0	0

Comments

Luca Compagna
The presentation "Towards D2 and D3: Initial set of results (06/07/2012)" is what was discussed in today (06/07/2012) meeting.
Jul 6, 2012 11:40 AM - Reply - Delete

Tools and specifications

Comments

Figure 4: SAP OAuth2 - Streamwork activity screenshot

All in all, this migration activity is progressing very well and the collaboration with the business unit is very fruitful and inspiring. In this respect, it is providing us a lot of insights about applying SPaCIoS methodology in industrial environments. One observation is that generating formal models from existing development artifacts needs to be investigated, since writing ASLan(++) models manually can be time-consuming and error-prone especially for non-expert users. Editing, debugging and simulation methods for ASLan(++) models would also help a lot in this respect. In addition, using the model checker as well as interpreting the obtained raw results can be challenging. In this respect, a very important aspect, together with abstract trace visualisation, is scalability. While it is clear that model checking of industrial size specification can be computational expensive in term of both time and memory, load-balancing can be offered to handle parallel model checking validations. Another observation is that a kind of "configuration management" of formal models and test-cases are needed for the following reasons:

- There are different options and potential settings in the security standard protocols being analyzed. A group of specifications exist correspondingly, which have common parts. When the common part of one specification is changed, we hope this change will be propagated correctly on all the others without repeating this effort manually.

- Testing sessions will be defined for combinations of (1) attack trace, (2) SUT definition, (3) message mapping, and (4) testing adapter. These session definitions need to be managed properly.

We also observed some challenges in generating and executing test-cases from abstract traces that were mainly due to mistakes between the message mapping and the testing adapter. These observations provide us ideas for potential features valuable for an industrial security testing platform powered by the SPaCIoS methodology. Some of these ideas have been already implemented as described in a specific paragraph below (see 3.1).

Last, but not least, this migration activity even triggered the initiation of another one about XML signature validation that is discussed hereafter.

Testing for XML Signature Wrapping Attack This migration activity originates from some discussions with SAP TIP Core SIM in the context of the Security Validation of SAP OAuth2 where XML Signature is also used. In particular the business unit asked whether we could use SPaCIoS methodology to also assess SAP XML Signature Validation software modules. The vulnerability-driven security testing approach of SPaCIoS resulted to be very suitable for this migration activity. In the vulnerability-driven security testing approach test cases are generated according to knowledge about specific attacks to perform penetration testing on web applications. In this migration activity, we migrate this technique to SAP to perform testing for XML Signature wrapping attack.

XML Signature is used to provide authentication and integrity of XML messages, e.g., Authentication Response in SAML SSO. XML Signature wrapping attack injects unauthorized data into a signed XML document alongside a possible restructuring in a way that the document's integrity is still verified. SAP has implementations of XML Signature validation on both Java and ABAP platforms. There is a need to perform testing to check whether SAP's implementations are vulnerable to XML Signature wrapping attack. We apply the vulnerability-driven security testing approach of SPaCIoS to generate test cases, which could be used by colleagues in business unit to test their own implementations. In this sense, this activity fits the "in-between" migration mode.

SPaCIoS Security Testing Environment at SAP The target of this migration activity, previously referred to as "TTCN-3 based Testing Environment", is to design a security testing environment to be used by SAP security analysts. When it is completed, it will contain some components of the SPaCIoS Tool developed in WP4, and some other components developed by SAP in WP6. This activity belongs to the "domain-specific tools" migration mode.

In an industrial environment, regression testing is needed, besides many other things, in order to make sure the already tested features or aspects are not affected when an application is updated. Thus, the ability of describing and storing test cases is needed in such a testing environment. Correspondingly, a test case description language can be very valuable. We considered to use TTCN-3, which is an international standard developed by ETSI, as the test case description language used in this testing platform.

Other observations emerging from the other industry migration activities at SAP were considered in the architecture of the SPaCIoS Security Testing Environment at SAP that, for the time being, is mainly leveraging on and enriching the Instrumentation-Based Testing Approach (see Section 4.2 of Deliverable 2.1.2, [6]). Three main phases can be identified in the Instrumentation-Based Testing Approach: specification (S), validation (V), and testing (T). Table 1 summarises what we have done to tackle some of these observations and to which phase of the Instrumentation-Based Testing Approach these features contribute to. One of these feature has been developed by SAP in the context of WP4. Below we provide more details about the new features that were implemented in the context of WP6.

#	Observation	Feature	S	V	T
F1	formal models generation		x		
F2	formal models enriched editing	ASLan++4Eclipse (WP4)	x		
F3	formal models debugging and simulation		x		
F4	model checker result interpretation	Inspecting the abstract trace (WP4 and WP6)		x	
F5	model checker execution and load-balancing	Preferences and run-configuration for model checking (WP6)		x	
F6	configuration management for formal models	SPaCIoS Navigator tag-based (WP6)	x	x	
F7	mismatch between message mapping and testing adapter	mismatch detector when defining the IUT and before executing test-case (WP6)			x
F8	management for test-cases generation and execution	Test campaign (WP6)			x
F9	regression testing	TTCN-3 based Testing Environment (WP6)			x

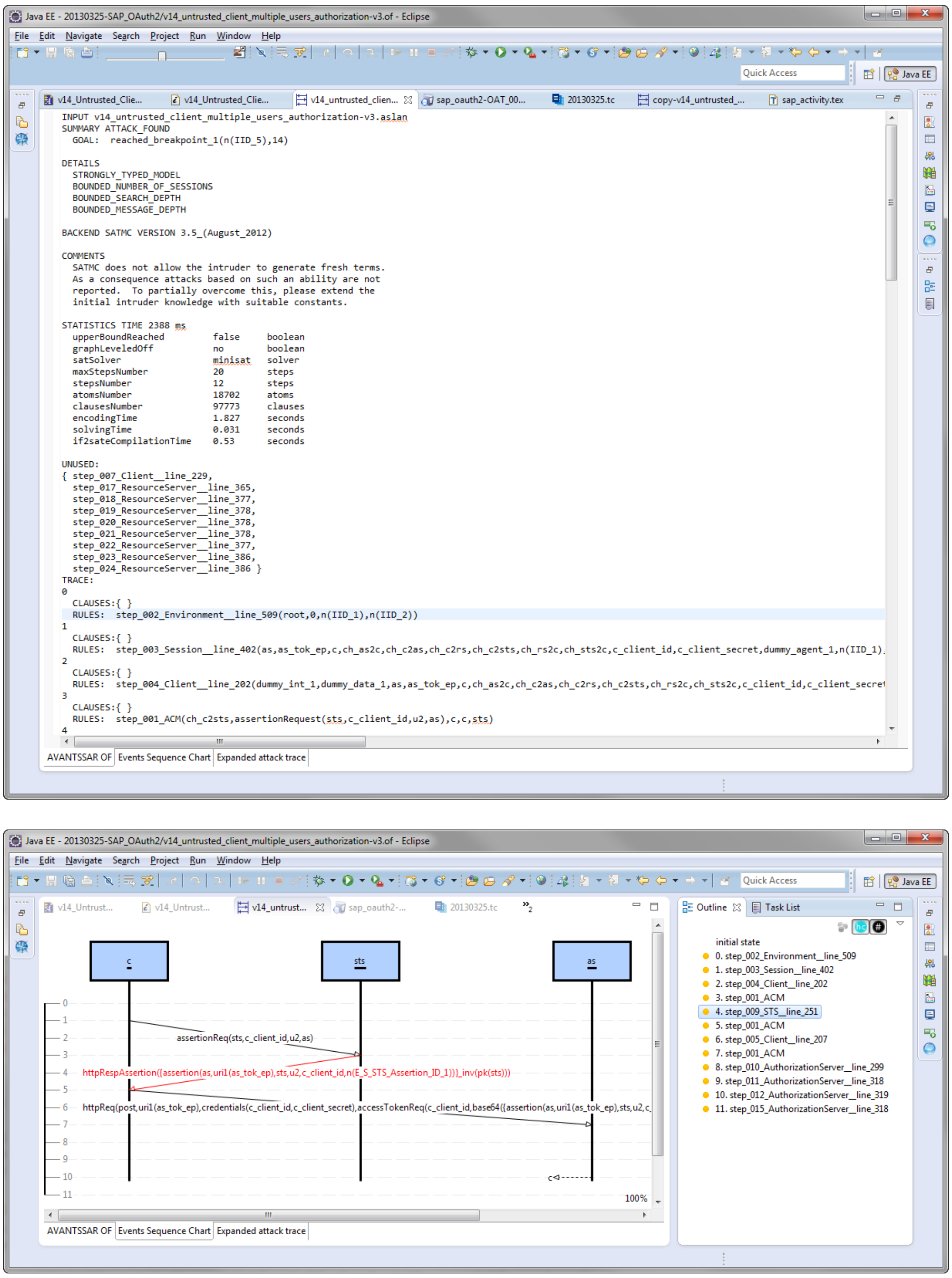
Table 1: Observations and enhancements.

- *F4 - Inspecting the abstract traces (WP4 and WP6).* This feature has been implemented in the context of both WP4 and WP6. Figure 5 depicts an example of how the raw result of the model checker (see top-part) is visualized to the tester (see bottom-part). The ASLan rules belonging to the abstract trace are summarized in the Outline and captured as send and receive events in the Event Message Chart (EMC). The tester can inspect the details of the abstract trace. For instance, by clicking on an ASLan rule name in the Outline the corresponding event will be presented in red on the EMC so to allow the tester to follow step-by-step the events of the abstract trace. Similarly by double-clicking on an ASLan rule in the Outline the ASLan will be opened showing the details of the rule including its body.

- *F5 - Preferences and run-configuration for model checking (WP6).* This feature has been implemented in the context of WP6. Figure 6 present the Advanced Model Checking Interface that we used in order to set the preferences for the model checkers to be used for a validation/testing campaign. Multiple model-checker entries can be specified and specific options can be set including the number of model-checking instances that can be run in parallel. At the moment local model checkers are supported, but this can be extended to run remote web services wrapping model-checker functionality.

- *F6 - SPaCIoS Navigator tag-based (WP6).* This feature has been implemented in the context of WP6. SAP SPaCIoS Navigator provides configuration management capability for formal models and derived abstract traces. When testing industrial scenarios like those underlying the SAP OAuth2 solution, the tester end-user faces the challenge to handle several variants of the formal models capturing these scenarios. Keeping track of the differences and commonalities between these variants becomes very important and can save a lot of time to the tester end-user. Figure 7 shows the main differences between the standard project explorer where formal model variants are presented as a simple list (cf. right-hand side) and the SAP SPaCIoS Navigator where these same variants are presented in a derivation hierarchy and enriched with tags that help the tester end-user to remember the peculiarities of that specific variant (cf. left-hand side). For instance, a quick look in the SAP SPaCIoS Navigator allows the tester to get many information about variant `v17` (for simplicity we do not write the entire name of the ASLan++ specification) without even opening it:

  - it has three tags to remember that in this variant (i) AS does not



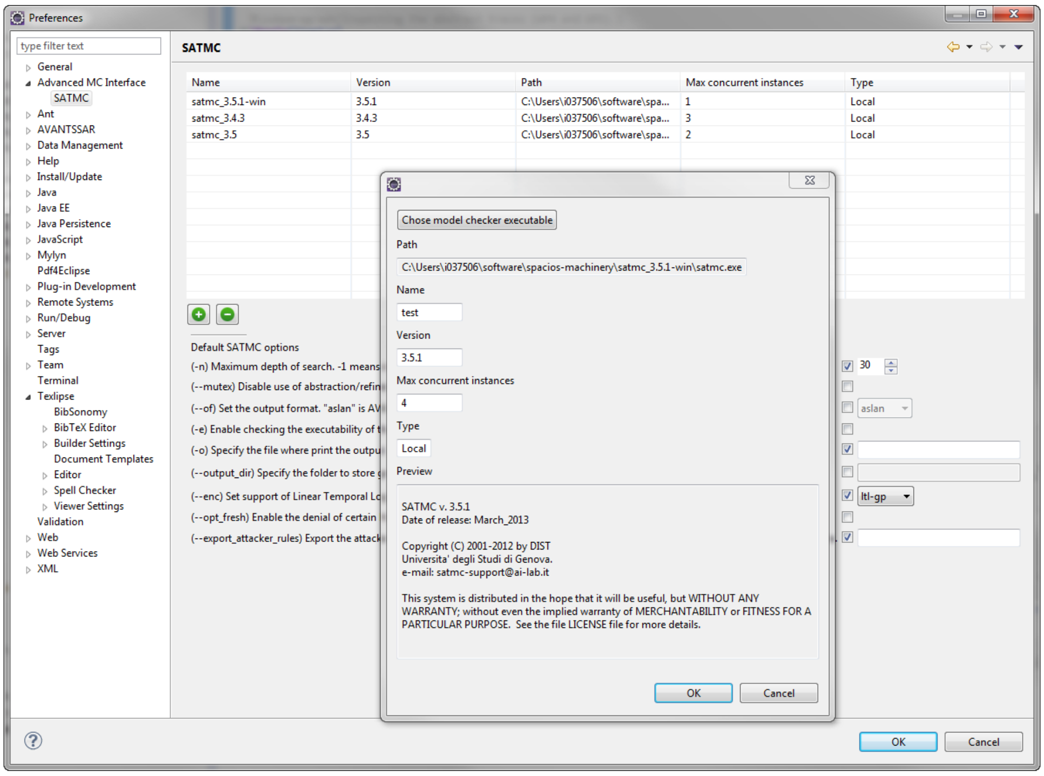
```
INPUT v14_untrusted_client_multiple_users_authorization-v3.aslan
SUMMARY ATTACK_FOUND
  GOAL:  reached_breakpoint_1(n(IID_5),14)

DETAILS
  STRONGLY_TYPED_MODEL
  BOUNDED_NUMBER_OF_SESSIONS
  BOUNDED_SEARCH_DEPTH
  BOUNDED_MESSAGE_DEPTH

BACKEND SATMC VERSION 3.5_(August_2012)

COMMENTS
  SATMC does not allow the intruder to generate fresh terms.
  As a consequence attacks based on such an ability are not
  reported.  To partially overcome this, please extend the
  initial intruder knowledge with suitable constants.

STATISTICS TIME 2388 ms
  upperBoundReached         false     boolean
  graphLeveledOff           no        boolean
  satSolver                 minisat   solver
  maxStepsNumber            20        steps
  stepsNumber               12        steps
  atomsNumber               18702     atoms
  clausesNumber             97773     clauses
  encodingTime              1.827     seconds
  solvingTime               0.031     seconds
  if2sateCompilationTime    0.53      seconds

UNUSED:
{ step_007_Client__line_229,
  step_017_ResourceServer__line_365,
  step_018_ResourceServer__line_377,
  step_019_ResourceServer__line_378,
  step_020_ResourceServer__line_378,
  step_021_ResourceServer__line_378,
  step_022_ResourceServer__line_377,
  step_023_ResourceServer__line_386,
  step_024_ResourceServer__line_386 }
TRACE:
0
  CLAUSES:{ }
  RULES:  step_002_Environment__line_509(root,0,n(IID_1),n(IID_2))
1
  CLAUSES:{ }
  RULES:  step_003_Session__line_402(as,as_tok_ep,c,ch_as2c,ch_c2as,ch_c2rs,ch_c2sts,ch_rs2c,ch_sts2c,c_client_id,c_client_secret,dummy_agent_1,n(IID_1)
2
  CLAUSES:{ }
  RULES:  step_004_Client__line_202(dummy_int_1,dummy_data_1,as,as_tok_ep,c,ch_as2c,ch_c2as,ch_c2rs,ch_c2sts,ch_rs2c,ch_sts2c,c_client_id,c_client_secret
3
  CLAUSES:{ }
  RULES:  step_001_ACM(ch_c2sts,assertionRequest(sts,c_client_id,u2,as),c,c,sts)
4
```

```
0
1
2   assertionReq(sts,c_client_id,u2,as)
3
4   httpRespAssertion({assertion(as,uri1(as_tok_ep),sts,u2,c_client_id,n(E_S_STS_Assertion_ID_1))}_inv(pk(sts)))
5
6   httpReq(post,uri1(as_tok_ep),credentials(c_client_id,c_client_secret),accessTokenReq(c_client_id,base64({assertion(as,uri1(as_tok_ep),sts,u2,c,
7
8
9
10                                                                                     c<-------
11
```

initial state
- 0. step_002_Environment__line_509
- 1. step_003_Session__line_402
- 2. step_004_Client__line_202
- 3. step_001_ACM
- 4. step_009_STS__line_251
- 5. step_001_ACM
- 6. step_005_Client__line_207
- 7. step_001_ACM
- 8. step_010_AuthorizationServer__line_299
- 9. step_011_AuthorizationServer__line_318
- 10. step_012_AuthorizationServer__line_319
- 11. step_015_AuthorizationServer__line_318

Figure 5: User-friendly visualization of abstract traces

authenticate C, (ii) C gets Access Token Response from AS, and



Name	Version	Path	Max concurrent instances	Type
satmc_3.5.1-win	3.5.1	C:\Users\i037506\software\spa...	1	Local
satmc_3.4.3	3.4.3	C:\Users\i037506\software\spa...	3	Local
satmc_3.5	3.5	C:\Users\i037506\software\spa...	2	Local

Default SATMC options

(-n) Maximum depth of search. -1 means	✓	30
(--mutex) Disable use of abstraction/refin		
(--of) Set the output format. "aslan" is AV		aslan
(-e) Enable checking the executability of t		
(-o) Specify the file where print the outpu	✓	
(--output_dir) Specify the folder to store g		
(--enc) Set support of Linear Temporal Lo	✓	ltl-gp
(--opt_fresh) Enable the denial of certain		
(--export_attacker_rules) Export the attack	✓	

Path
C:\Users\i037506\software\spacios-machinery\satmc_3.5.1-win\satmc.exe
Name
test
Version
3.5.1
Max concurrent instances
4
Type
Local
Preview

```
SATMC v. 3.5.1
Date of release: March_2013

Copyright (C) 2001-2012 by DIST
Universita' degli Studi di Genova.
e-mail: satmc-support@ai-lab.it

This system is distributed in the hope that it will be useful, but WITHOUT ANY
WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.  See the file LICENSE file for more details.
```

Figure 6: Advanced Model Checking Interface

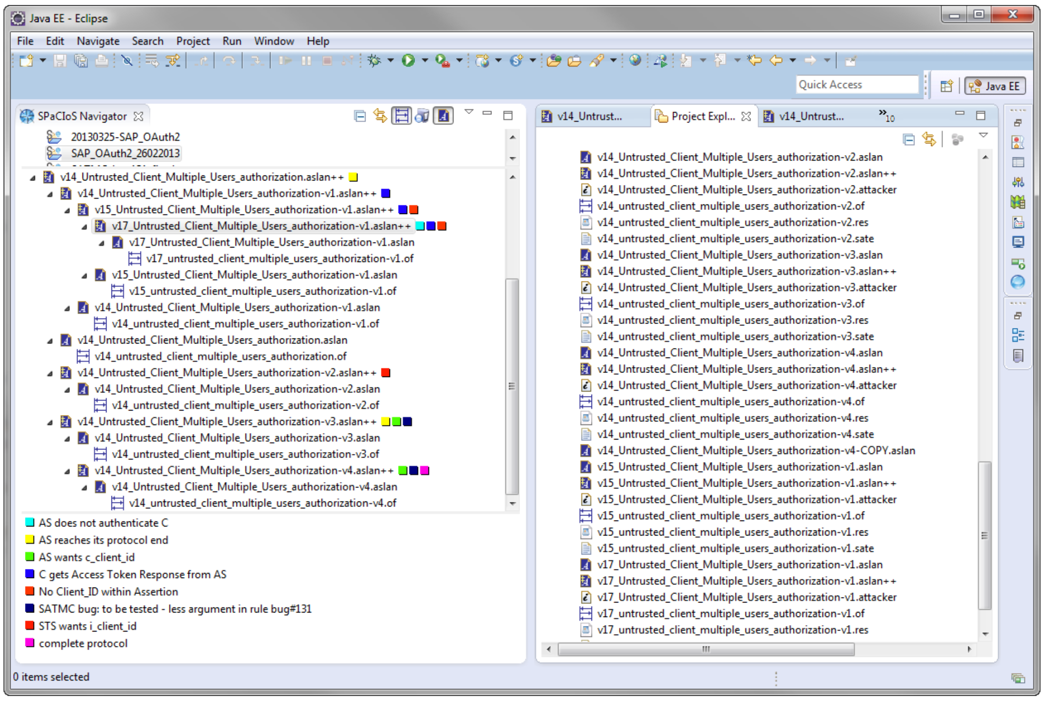
(iii) the `Client_ID` is not within the Assertion;

– an abstract trace has been found for the ASLan model associated to this variant; and

– it derives directly from `v15` that in turn derives from `v14` and so on and so forth. (A simple diff is now sufficient to understand in details the differences between two related variants.)

- *F7 - Mismatch detector when defining the IUT and before executing the test-case (WP6).* This feature has been implemented in the context of WP6. When preparing for test-case generation and execution the tester has to specify the IUT that comprises the message mapping and to code the adapter that will execute the construction and parsing of the messages exchanged in the test case execution. In doing so the tester has to follow certain conventions and mistakes can be made. Figure 8 shows an example (black rectangles have been manually added to hide sensible information and they should be ignored):

  – the ASLan symbol `httpRequest` is mapped to an Adapter Class



```
20130325-SAP_OAuth2
SAP_OAuth2_26022013
v14_Untrusted_Client_Multiple_Users_authorization.aslan++ 🟨
  v14_Untrusted_Client_Multiple_Users_authorization-v1.aslan++ 🟦
    v15_Untrusted_Client_Multiple_Users_authorization-v1.aslan++ 🟦🟥
      v17_Untrusted_Client_Multiple_Users_authorization-v1.aslan++ 🟦🟦🟥
        v17_Untrusted_Client_Multiple_Users_authorization-v1.aslan
          v17_untrusted_client_multiple_users_authorization-v1.of
      v15_Untrusted_Client_Multiple_Users_authorization-v1.aslan
        v15_untrusted_client_multiple_users_authorization-v1.of
    v14_Untrusted_Client_Multiple_Users_authorization-v1.aslan
      v14_untrusted_client_multiple_users_authorization-v1.of
  v14_Untrusted_Client_Multiple_Users_authorization.aslan
    v14_untrusted_client_multiple_users_authorization.of
  v14_Untrusted_Client_Multiple_Users_authorization-v2.aslan++ 🟥
    v14_Untrusted_Client_Multiple_Users_authorization-v2.aslan
      v14_untrusted_client_multiple_users_authorization-v2.of
  v14_Untrusted_Client_Multiple_Users_authorization-v3.aslan++ 🟨🟩🟦
    v14_Untrusted_Client_Multiple_Users_authorization-v3.aslan
      v14_untrusted_client_multiple_users_authorization-v3.of
    v14_Untrusted_Client_Multiple_Users_authorization-v4.aslan++ 🟩🟦🟪
      v14_Untrusted_Client_Multiple_Users_authorization-v4.aslan
        v14_untrusted_client_multiple_users_authorization-v4.of
```

🟦 AS does not authenticate C
🟨 AS reaches its protocol end
🟩 AS wants c_client_id
🟦 C gets Access Token Response from AS
🟥 No Client_ID within Assertion
🟦 SATMC bug: to be tested - less argument in rule bug#131
🟥 STS wants i_client_id
🟪 complete protocol

```
v14_Untrusted_Client_Multiple_Users_authorization-v2.aslan
v14_Untrusted_Client_Multiple_Users_authorization-v2.aslan++
v14_Untrusted_Client_Multiple_Users_authorization-v2.attacker
v14_untrusted_client_multiple_users_authorization-v2.of
v14_untrusted_client_multiple_users_authorization-v2.res
v14_untrusted_client_multiple_users_authorization-v2.sate
v14_Untrusted_Client_Multiple_Users_authorization-v3.aslan
v14_Untrusted_Client_Multiple_Users_authorization-v3.aslan++
v14_Untrusted_Client_Multiple_Users_authorization-v3.attacker
v14_untrusted_client_multiple_users_authorization-v3.of
v14_untrusted_client_multiple_users_authorization-v3.res
v14_untrusted_client_multiple_users_authorization-v3.sate
v14_Untrusted_Client_Multiple_Users_authorization-v4.aslan
v14_Untrusted_Client_Multiple_Users_authorization-v4.aslan++
v14_Untrusted_Client_Multiple_Users_authorization-v4.attacker
v14_untrusted_client_multiple_users_authorization-v4.of
v14_untrusted_client_multiple_users_authorization-v4.res
v14_untrusted_client_multiple_users_authorization-v4.sate
v14_Untrusted_Client_Multiple_Users_authorization-v4-COPY.aslan
v15_Untrusted_Client_Multiple_Users_authorization-v1.aslan
v15_Untrusted_Client_Multiple_Users_authorization-v1.aslan++
v15_Untrusted_Client_Multiple_Users_authorization-v1.attacker
v15_untrusted_client_multiple_users_authorization-v1.of
v15_untrusted_client_multiple_users_authorization-v1.res
v15_untrusted_client_multiple_users_authorization-v1.sate
v17_Untrusted_Client_Multiple_Users_authorization-v1.aslan
v17_Untrusted_Client_Multiple_Users_authorization-v1.aslan++
v17_Untrusted_Client_Multiple_Users_authorization-v1.attacker
v17_untrusted_client_multiple_users_authorization-v1.of
v17_untrusted_client_multiple_users_authorization-v1.res
```

0 items selected

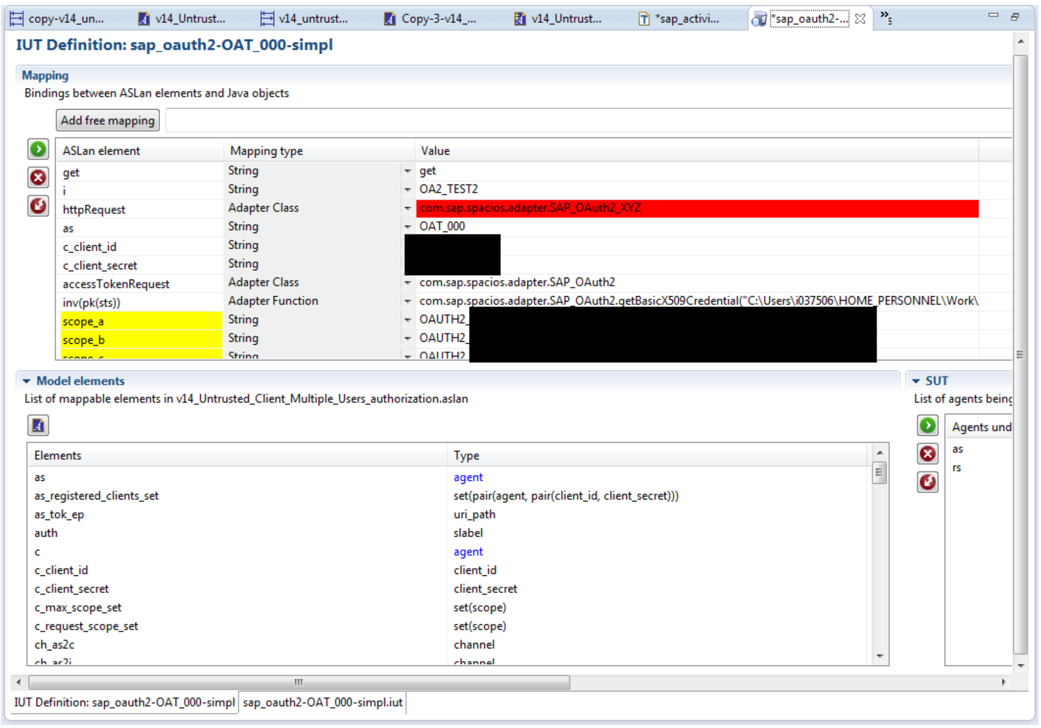
Figure 7: SAP SPaCIoS Navigator

that does not exists. This is properly detected and highlighted in red as the test-case generation and execution would clearly fail.

– the symbols `scope_a`, `scope_b`, . . . do not belong to the ASLan specification. While this does not endanger the test-case generation and execution and it could be done in purpose, still the tester is warned in that respect through a yellow highlight.

All in all, these simple checks can save a lot of time during test-campaign. Even more important, a similar check is executed just before execution of a test-case when all the information available can be matched to detect mistakes that would prevent the execution itself. For instance, all the ASLan symbols occurring in sending and receiving messages of the abstract trace that involve entities outside the system under testing (SUT) must have a corresponding mapping within the IUT and this mapping shall be properly implemented within the adapter. Any mistake detected via these checks will be listed together with quick fixes to the tester end-user.

- *F8 - Test campaign (WP6).* This feature has been implemented in the context of WP6. When testing industrial scenarios like those underly-



IUT Definition: sap_oauth2-OAT_000-simpl

Mapping

Bindings between ASLan elements and Java objects

ASLan element	Mapping type	Value
get	String	get
i	String	OA2_TEST2
httpRequest	Adapter Class	com.sap.spacios.adapter.SAP_OAuth2_XYZ
as	String	OAT_000
c_client_id	String	[redacted]
c_client_secret	String	[redacted]
accessTokenRequest	Adapter Class	com.sap.spacios.adapter.SAP_OAuth2
inv(pk(sts))	Adapter Function	com.sap.spacios.adapter.SAP_OAuth2.getBasicX509Credential("C:\Users\i037506\HOME_PERSONNEL\Work\
scope_a	String	OAUTH2_[redacted]
scope_b	String	OAUTH2_[redacted]
scope_c	String	OAUTH2[redacted]

Model elements

List of mappable elements in v14_Untrusted_Client_Multiple_Users_authorization.aslan

Elements	Type
as	agent
as_registered_clients_set	set(pair(agent, pair(client_id, client_secret)))
as_tok_ep	uri_path
auth	slabel
c	agent
c_client_id	client_id
c_client_secret	client_secret
c_max_scope_set	set(scope)
c_request_scope_set	set(scope)
ch_as2c	channel
ch_ar2i	channel

SUT

List of agents being

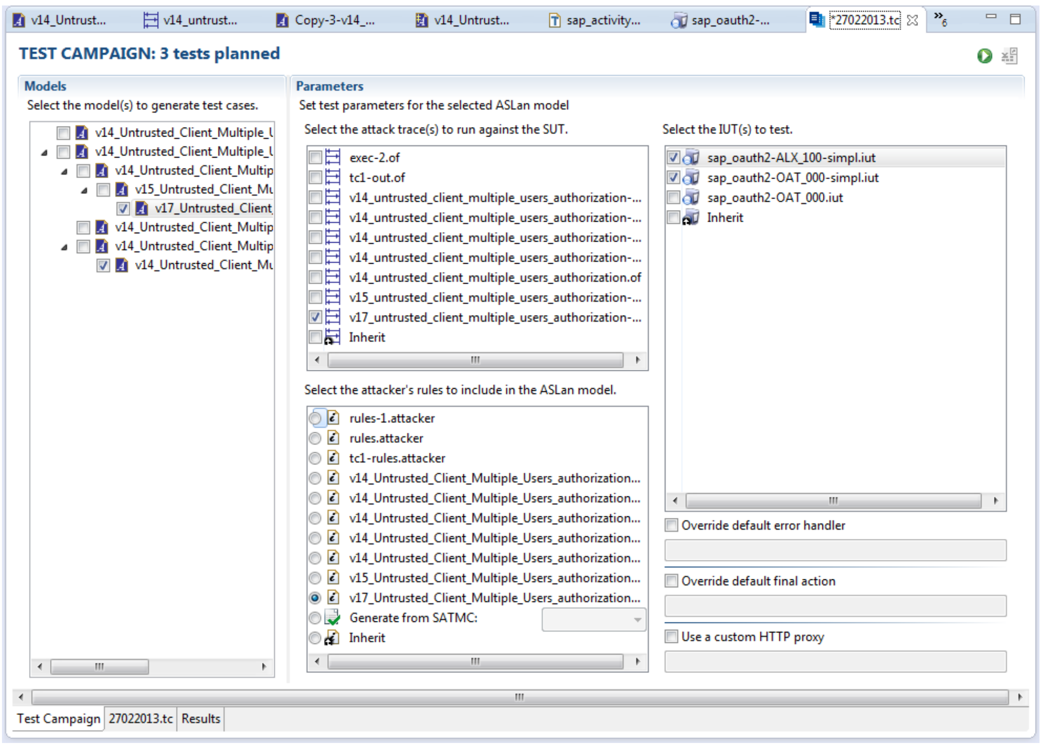
Agents und
as
rs

IUT Definition: sap_oauth2-OAT_000-simpl
sap_oauth2-OAT_000-simpl.iut

Figure 8: IUT definition - mismatch detector

ing the SAP OAuth2 solution, the tester end-user faces the challenge to handle several test-cases executions of abstract traces derived from the many formal model variants and perhaps on several IUTs. The Test Campaign plug-in, depicted in Figure 9, supports the tester end-user in this respect. Multiple formal model variants can be selected and for each of them is possible to specify which abstract traces shall be executed and against which IUTs. In the example of Figure 9, the tester has selected both `v17` and the `v14` at the bottom (for simplicity we do not write the entire name of the ASLan++ specifications and abstract traces). For what concerns `v17` the corresponding abstract trace `v17` shall be executed against `sap_oauth2-ALX_100-simpl.iut` and `sap_oauth2-OAT_000-simpl.iut`, two IUTs representing SAP systems deploying the SAP OAuth2 solution. By pressing the play button all the test-cases defined in the testing campaign are executed one after the other (parallelism could be envisaged) and result details including exchanged HTTP requests and responses are properly stored for later access.

- *F9 - TTCN-3 based Testing Environment (WP6).* This feature has


v14_Untrust... | v14_untrust... | Copy-3-v14_... | v14_Untrust... | sap_activity... | sap_oauth2-... | *27022013.tc

TEST CAMPAIGN: 3 tests planned

Models
Select the model(s) to generate test cases.

- v14_Untrusted_Client_Multiple_U
- v14_Untrusted_Client_Multiple_U
  - v14_Untrusted_Client_Multip
    - v15_Untrusted_Client_Mu
      - [✓] v17_Untrusted_Client
  - v14_Untrusted_Client_Multip
  - v14_Untrusted_Client_Multip
    - [✓] v14_Untrusted_Client_Mu

Parameters
Set test parameters for the selected ASLan model

Select the attack trace(s) to run against the SUT.

- exec-2.of
- tc1-out.of
- v14_untrusted_client_multiple_users_authorization-...
- v14_untrusted_client_multiple_users_authorization-...
- v14_untrusted_client_multiple_users_authorization-...
- v14_untrusted_client_multiple_users_authorization-...
- v14_untrusted_client_multiple_users_authorization.of
- v15_untrusted_client_multiple_users_authorization-...
- [✓] v17_untrusted_client_multiple_users_authorization-...
- Inherit

Select the attacker's rules to include in the ASLan model.

- rules-1.attacker
- rules.attacker
- tc1-rules.attacker
- v14_Untrusted_Client_Multiple_Users_authorization...
- v14_Untrusted_Client_Multiple_Users_authorization...
- v14_Untrusted_Client_Multiple_Users_authorization...
- v14_Untrusted_Client_Multiple_Users_authorization...
- v14_Untrusted_Client_Multiple_Users_authorization...
- v15_Untrusted_Client_Multiple_Users_authorization...
- [✓] v17_Untrusted_Client_Multiple_Users_authorization...
- Generate from SATMC:
- Inherit

Select the IUT(s) to test.

- [✓] sap_oauth2-ALX_100-simpl.iut
- [✓] sap_oauth2-OAT_000-simpl.iut
- sap_oauth2-OAT_000.iut
- Inherit

Override default error handler

Override default final action

Use a custom HTTP proxy

Test Campaign | 27022013.tc | Results

Figure 9: Test campaign

been implemented in the context of WP6. It is based on TTCN-3, which is an international standard developed by ETSI, as the test case description language. In a testing environment based on TTCN-3, in addition to an interpreter of the TTCN-3 language itself, there needs to be an adapter to map the TTCN-3 `send()` and `receive()` to concrete message sending and receiving. We developed a TTCN-3 test case interpreter, which supports a subset of the standard TTCN-3 language, and an HTTP adapter which is able to create/process SAML messages. The adapter was successfully used in testing real implementation of SAML. This feature is decoupled from the others and provide a different test execution engine. One possible direction for integrating this feature with the others is translation of Java instrumented testcases into TTCN-3 instrumented ones.

Model Generation In order to pave the way to the "domain-specific tools" migration mode, we investigate the possibility of generating formal models in ASLan(++) in an automatic or semi-automatic way. We consider the following possible scenarios:

- Generating ASLan(++) model from MSC. MSC is used in application design at SAP. They describe the message passing between entities in an application, and capture aspects relevant for security analysis, such as key fields in messages. According to our experiences, ASLan++ models generated from MSCs are on the right level of abstraction, and useful for the security validation and testing. We have a quite precise idea on how to implement this feature and we are currently working on a feasibility study on top of necessary effort, available resources, and cost-benefit in general. We also believe the result of this initiative could be adapted to make them effective also outside the SAP premises. For instance, standardization bodies could leverage on this during the drafting of their security standards and get immediate advantage of SPaCIoS machinery without being experts in formal methods.

- Generating ASLan(++) model from ABAP source code. ABAP is a programming language widely used at SAP. It has its own syntax, programming framework, and runtime framework. If we are able to generate pieces of ASLan(++) model from ABAP source code, the task of application modeling could be made easier. One way we are investigating is to generate Control Flow Graph (CFG) from ABAP source code, and use the white-box model inference tool developed by IeAT to create ASLan(++) model from the CFG. One the other hand, generating a proper formal model from ABAP for security analysis purpose is not straightforward, e.g., we need to deal with statements accessing database, and performing numeric calculations.

- Generating/linking ASLan(++) model from/to USDL. We have been investigating more on USDL (recently recast upon the Linked Data principle to better promote and support the use of USDL on the Web, see [10]) and its USDL-SEC module to understand whether ASLan(++) models could be generated from them or at least linked to them. As we expected linking an ASLan(++) model to USDL is definitely possible. For what concerns model generation, while it is possible to link to a USDL specification a BPMN specification of a composed service so to represent with standardized established notations service composition and orchestration, it does not seem viable to use USDL-SEC to capture the formal security property that this composition should guarantee. In fact, USDL-SEC offers a way to describe the security goals that the service claims to guarantee for business users, but the abstraction level is too high to trigger validation with the SPaCIoS machinery. For instance, USDL-SEC allows to say that the service on the market-place guarantees confidentiality via encryption with 2048 length-key, but it does not say which assets are protected, against who, and how. It is worth noticing that this should not be read as a critic towards USDL-SEC. USDL-SEC was devised with the ultimate goal of conveying security-relevant information of the service to a business user and this is definitely achieved. The very similar challenge towards USDL-SEC has been faced by the Assert4SOA EU project that is devising its own language for describing the *assert* of the service. Such a language could be useful also to capture some of the security properties suitable to trigger the validation via SPaCIoS, but this requires more investigation.

Penetration Testing Strategy In SAP, Technical Validation is the business unit responsible for security testing in SAP. Currently, we collaborate with colleagues in Technical Validation to create a penetration testing strategy document for SAP. The parts we are contributing include:

- Testing planning based on attack tree models. In the attack tree model for a specific application, each node is labeled with metrics such as resource needed, cost, probability, etc. Based on these metrics, attack scenarios will be prioritized. Testing activities will be planned accordingly.
- Performing penetration testing based on low level attacker models. For each specific node, which is a specific attack, in an attack scenario, penetration testing are performed based on low level attacker model of this attack.

This under-development strategy is specific to SAP, because SAP's applications which we want to test have their own specific features. In the environment aspects, they have SAP-specific programming, design, and run-time environments. In the procedure aspects, design artifacts existing in several forms are accessible. All these facts need to be taken into consideration.

This industrial activity does not map directly to the three mode we identified, but it will have a long term impact to SAP's business and the experience we are gathering in working on the SPaCIoS project is providing valuable insights to contribute here.

3.2 Siemens' activities

In the second and third years of the project, Siemens has continued the process described in the previous Deliverable 6.2.1 [5] in the attempt of identifying successful scenarios for the migration of the SPaCIoS technology to its operation. In particular, as it has been described in both Deliverables 6.2.1 [5] and 6.1.2 [4], the wide range of industry sectors within Siemens makes a unique high-level specification formalism in the context of security impractical: due to the different business activities of Siemens in very different areas ranging from factory automation over SCADA systems for power grids to health care appliances and other areas, there is no single, uniform approach to security modeling, analysis, and testing. To cope with this complexity, Siemens deploys the model of a consultancy service for all security related issues within its Corporate Technology unit, including, in particular, a security vulnerability testing approach.

A centralized approach has been proposed: The Siemens Corporate Technology is able to provide a security organization for the different Siemens sectors. By implementing the lessons learned in two key security research groups within the Siemens Corporate Technology, ITS CSA (Corporate Security Assessments) and ITS SEA (Security Architecture), it is possible to apply the results from SPaCIoS in a wide range of Siemens applications. This makes the migration within these groups to be fundamental.

Furthermore, Siemens Corporate Technology with 6000 employees is also an altogether unrealistic target. It is important to identify key groups within Corporate Technology that can benefit from the project's technology and prototypes. In Year 2 and in the first months of Year 3, as the tools are beginning to be more mature and the processes well-understood, it has become also clearer that the greatest beneficiary is the CERT Security Assessments (CSA) group within the IT Security technology field. In fact, not only the activities in CERT can be enriched by the insights of the project, but also they can serve as a guide when it comes to understand how to bridge the gap between theory and practice in the context of SPaCIoS technologies.

What is CERT Security Assessments? CERT Security Assessments (CSA), one of the groups within the IT Security field, focuses in particular on the challenging task of testing Siemens products for known vulnerabilities. This task is particularly challenging because on the one hand it implies a serious, committed effort for being up-to-date with the latest exploits and hacking techniques, and therefore requires its employees to be in constant contact with the penetration test community and academic research on the field. On the other hand, many of the testing activities are performed in a black-box fashion, to simulate real attackers. This means that techniques that can improve the rigor and the coverage of interesting test cases are likely to have a positive impact within this group. Currently CERT performs almost 100 security assessments of applications and products per year, a number that has consistently increased over the past years.

Vulnerability-testing in CERT. To maximize the time invested in assessing a product, the security experts in CERT follow a detailed list of known vulnerabilities as base for their testing activities. To keep this list up-to-date, the security pen-testers meet weekly to discuss vulnerability and assessment issues and to present the new threats that have been observed or have been discussed or documented in white-hat forums. In an annual meeting, they revise the complete list, looking carefully at the type of attacker capabilities or resources needed. This is very much in line with the vulnerability driven-testing described in Deliverable 2.4.1 [7], but with the disadvantage that much of the testing relies on (a) commercially available software with hard-coded attacker models and (b) personal experience and white-hacking skills. Therefore one of the potentially more interesting outcomes of industry migration to Siemens is a standardized language for describing attacker models and libraries of state-of-the-art attacks that can be systematically used by the penetration testers. Of course, we do not intend to replace entirely the current interactive penetration testing methodology at Siemens, nor to make it fully automatic, but to improve it with new methods and tools.

To summarize the main advantages of applying the SPaCIoS technology in this area with respect to the state of the art are among others:

- **More automation.** Current penetration testing (besides automatic fuzzing) offers little automation support. The expert repeats a loop where he is always taking new decisions: he sends one HTTP message, observes the reaction, decides which step to perform next in a new message, etc. We expect to have long sequences of events happening automatically and the interaction with the expert to be less intense.
- **Test standardization.** For the purposes of testing, much of the required know-how is distributed in the heads of the experts. One expert is very good in finding cross-site scripting another one in testing for stack overflows, etc. We expect from a SPaCIoS methodology that the different experts are able to conduct more similar "standardized" tests (standardized at different assurance levels).
- **Decision support**. When the expert is asked to conduct the test he can count on a set of alternatives proposed by the tools, based on the type of software to be tested (as represented by the behavioral models or data-flow models, or as given by configuration files), and based on

the risk and threads that have been evaluated for the particular system under evaluation.

- **Increased rigor and precision.** Since by guiding the testing with formal models, it is less likely to ignore interfaces, event sequences or situations that are likely to be vulnerable to attacks.
- **Systematic walk-through over the application under test.** Model inference can help the tester to easily obtain (partial) those formal models, at least partially, and use that information to explore the security issues present in the application.
- **Flexibility and extendibility.** The extensible library of detailed vulnerability attack trees associated with vulnerabilities can be easily maintained and is not hardcoded in a pen-testing tool.

Modeling of risks and attackers in CERT. As it is the case with most IT solutions, a fundamental part of assessing security is understanding the risk and costs associated to worst case scenarios in particular contexts. To be able to judge those risks, CSA conducts thorough interviews with product owners and workshops with both technology and business related contacts. These result in estimations about the so-called Security Attestation Level (SAL), which determines in which depth and which vulnerabilities should be taken into account to perform the analysis. Currently, CERT Security Assessments works with SAL levels from 1 to 3 reflecting the skills, time and money an adversary would invest attacking the application. The security attestation level directly affects the kind of vulnerabilities that are tested, the depth of the test, as well as the amount of person days invested in a security assessment. If security issues are found in an application, a risk value is computed based on a number of parameters including the difficulty of exploiting the issue, the requirements to do so as well as the impact it would have in the productive system. Additionally possible solutions are presented. In particular, the attacker libraries obtained using the low-level attacker models of Deliverables 2.4.1 [7] and 3.3 [9] can be classified according to the security assessment levels used by CERT and in that way a standard set of attacks would be picked automatically according to the selected level. If a precise connection between those low-level attackers and the abstract models (as shown preliminary by the Spacite Tool) is achieved, then it is expected that in the presence of formal models of the applications we can obtain a better coverage of the interesting application points as opposed to black-box testing. In absence of models, the various model inference technologies proposed in the project represent an interesting automatic alternative to manual model generation. A deeper assessment of the technology delivered by SPaCIoS and its applications to CERT will be possible as long as the tools and methodologies progress further. For a preliminary assessment see Deliverable 5.2 [8].

All in all, our (Siemens) vision for a successful long run migration of the SPaCIoS methodologies, artifacts and tools is the *support of penetration testing* based on

- a formal model of the system: a model of the expected/intended behavior of a system can help testers to reach interesting states and interfaces that are not guaranteed to be reached with black-box testing
- the SPaCIoS technology: the set of tools, including model inference and various testing techniques based on formal models are very promising towards enriching the current penetration tester's tool-set
- a library of vulnerabilities and attack models: standardization on the attacker models and vulnerabilities to be tested guarantees consistent quality of tests
- specialized tools for low-level-vulnerability-driven test case generation, including in particular the Vera tool and the methodology for vulnerability-driven assessment workflow, described in detail in Deliverable 3.3 [9], Chapters 2 and 3.

3.2.1 Evaluation of the Migration to Siemens

We have already started to apply the SPaCIoS methodology, technology, and tools during real-world security assessments, and we plan to continuously apply them as they become mature.

Within this workpackage (WP6), we will evaluate the results of the industrial migration. For this purpose, we will develop a questionnaire to capture observations after each single step of selected assessments and thus support a systematic evaluation regarding benefits, drawbacks, practicability, areas for improvement, and inherent limitations. In this way, the value of the methods and tools will be evaluated for each step of typical assessments.

The following are the typical steps of a vulnerability assessment (some of which are optional, depending exactly on which tools or methodology is used):

1. Elicitation of high-level security requirements;

2. Creation of a technical system description in the form of a Data-Flow-Diagram (DFD);

3. Creation of a security overview, in the form of security annotations to the DFD;

4. Definition of the Business Worst Case Scenarios (BWCSs);

5. Identification of the concrete threats to the BWCSs, or Technical Threat Scenarios TTSs;

6. Mapping of TTSs to BWCSs, and evaluation of the impact of TTSs;

7. Determination of the likelihood of unwanted events, including the required estimated skill level for an attacker, and the risks associated to BWCSs and TTSs;

8. Prioritization of attacker models (say, in Vera) based on the risk analysis of the system under test (SUT);

9. Prioritization of parameters and instantiation libraries;

10. Instantiation of attacks with specific values;

11. Instantiation of configuration values;

12. Generation of attacker model from high-level (say, ASLan++) model;

13. Selection of attack libraries; and

14. Run of tests.

The following list contains an initial set of questions, which evaluate the support of the methodologies and tools during vulnerability assessment:

- What tools / methodology did you use in each step?
- Compared to the whole assessment, what is the proportion of time required for each step?
- For each step, did the effort of using the tools and method outweigh its benefit?
- For each step, what turned out to be easier or more challenging in practice than expected?

- What relevant technical information could not be obtained by the used tools?
- What additional information was included in the security overview, compared to the technical overview?
- Would dynamic aspects in the security overview have been useful for TTS elicitation or the rule-based derivation of test types?
- What was the proportion of rule-generated tests and manually selected tests?
- What was the proportion of tests from the library and custom "freestyle" tests?

The pen-testers will be also asked to answer questions that will help us to define objectives for future work:

- Develop a rule of thumb for how long each step should (relatively) take.
- Identify a minimal required time budget.
- Identify useful tools and increase automation.
- Decide if additional or different diagram types should be used.

The following list presents six requirements that the tests determined and run by the SPaCIoS methodology and tools should satisfy within the Siemens landscape:

R1 Each security test, alone or in combination with other tests, addresses at least one BWCS.

R2 Each security test targets the proper system element.

R3 Each security test aims at violating the right security property.

R4 Each security test has the proper level of sophistication with respect to the expected threat agents. That is, it corresponds to the assumed attacker capabilities (skill level).

R5 Each security test reflects the technology, implementation and configuration of the SUT.

R6 The priority of each security test is high enough with respect to the given time and budget.

4 Lessons Learned and Recommended Best Practices

In this section, we report the lessons learned from the migration activities and recommend best practices. The following bullet items summarize the preliminary lessons learned until now.

- *Setting up a win-win collaboration.* Signing up a collaboration with a business unit is not easy. Units have very strict deadlines and full backlog lists to deliver products and features to customers. In our experience, the keys to set up a win-win collaboration are to emphasize the low-hanging fruits as well as the long term vision, and to be clear in the transfer activity plan and objectives (redaction of a small internal project description of work with concrete deliverables and effort estimation) so to properly manage expectations.

- *Consultancy requires effort, but it is a good start.* A migration activity (even when running as pure consultancy) is not confined to the delivery of consultancy outcomes. Besides providing useful insights and feedback for adoption of research outcomes within industry, it also serves as a means to open new research and transfer directions and to start new collaborations.

- *Involve business unit as soon as possible and properly.* Starting from the collaboration proposal time, keep regular interactions with business units once the project starts. At the same time, it is important to keep their involvement limited, in order not to be perceived as a time-waster.

- *Be focused and execute.* There may be many opportunities for transferring research results. It is important to look-ahead, but properly balance the investigation effort and take decisions so to be capable of executing on the most promising ones and reach some concrete results there.

5 Conclusions

References

[1] OASIS. Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0. http://docs.oasis-open.org/security/saml/v2.0/samlbindings-2.0-os.pdf, March 2005.

[2] OASIS. Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0. http://docs.oasis-open.org/security/saml/v2.0/samlprofiles-2.0-os.pdf, March 2005.

[3] OASIS. SAML V2.0 Errata, May 2012. Available at: http://docs.oasis-open.org/security/saml/v2.0/errata05/os/saml-v2.0-errata05-os.html.

[4] SPaCIoS. Deliverable 6.1.2: Dissemination and Exploitation Plan v.1, 2011.

[5] SPaCIoS. Deliverable 6.2.1: Industrial service description language and security validation tools in IoS, 2011.

[6] SPaCIoS. Deliverable 2.1.2: Modeling security-relevant aspects in the IoS, 2012.

[7] SPaCIoS. Deliverable 2.4.1: Definition of Attacker Behavior Models, 2012.

[8] SPaCIoS. Deliverable 5.2: Proof of Concept and Tool Assessment v.2, 2012.

[9] SPaCIoS. Deliverable 3.3: SPaCIoS Methodology and technology for vulnerability-driven security testing, 2013.

[10] W3C USDL Incubator Group. Linked USDL. http://www.linked-usdl.org/, 2012.