SPaCIoS

Secure Provision and Consumption in the Internet of Services

STREP Project number: 257876

Objective ICT-2009.1.4 c: Technology and Tools for Trustworthy ICT

01 Oct 2010 − 30 Sep 2013

www.spacios.eu

Alexander Pretschner, POFI 2011, Pisa, June 8th, 2011
iww J. Oudinet, M. Büchler, SPACIOS consortium

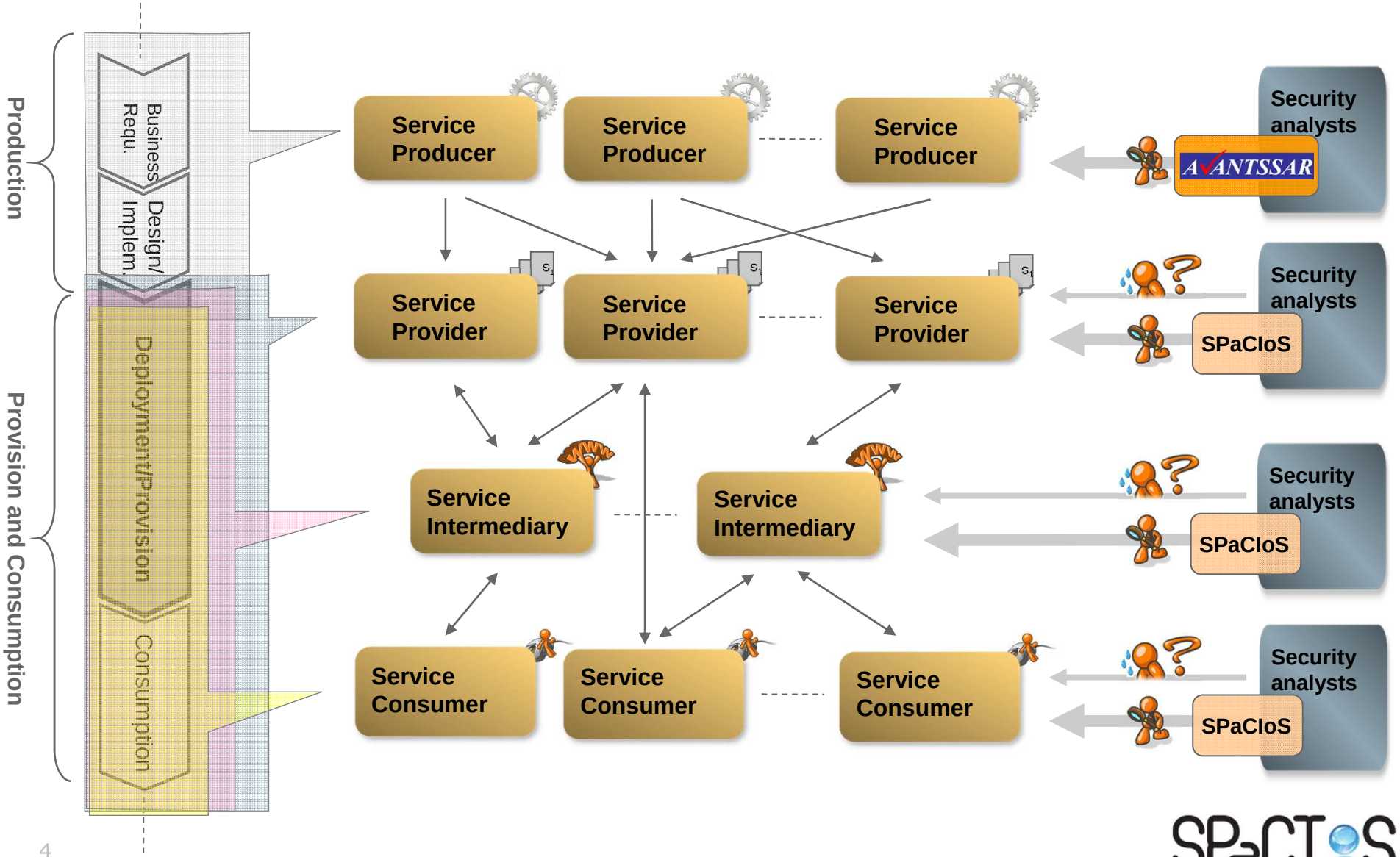
Agenda

- Motivation
- Consortium
- Project structure
- Resarch Core (KIT perspective)
- Conclusions

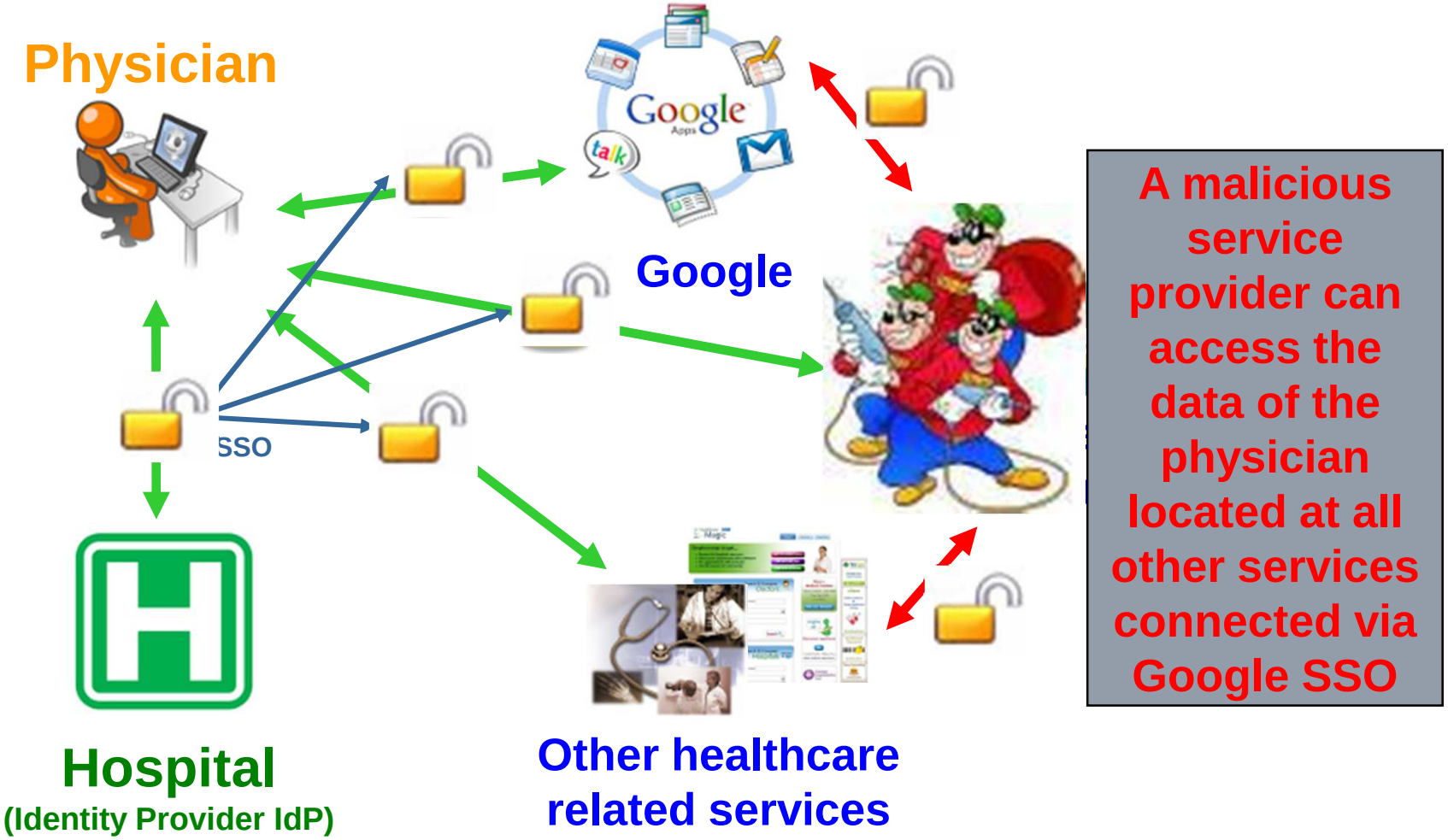
Services and their main stakeholders

- **Services provide business functionalities**
  - Business functionalities are typically the result of composing distributed services
  - Services include web-based applications and web services
  - Services do not imply any specific technology for implementing them
- **Four main service stakeholders (having their own security requirements)**
  - **Producers**: design and implement services
  - **Providers**: provide and deploy services
  - **Consumers**: consume services at runtime
  - **Intermediaries**: provide services to consumers by collecting services by providers
    - have specific trust relationships with consumers and providers
    - e.g., service brokers, service aggregators, etc
- **State of the art**
  - **Security analysts**: offer consultancy to stakeholders to analyze security requirements
  - **Existing tools**: not integrated, not based on a scientific approach, not at all automated, etc.

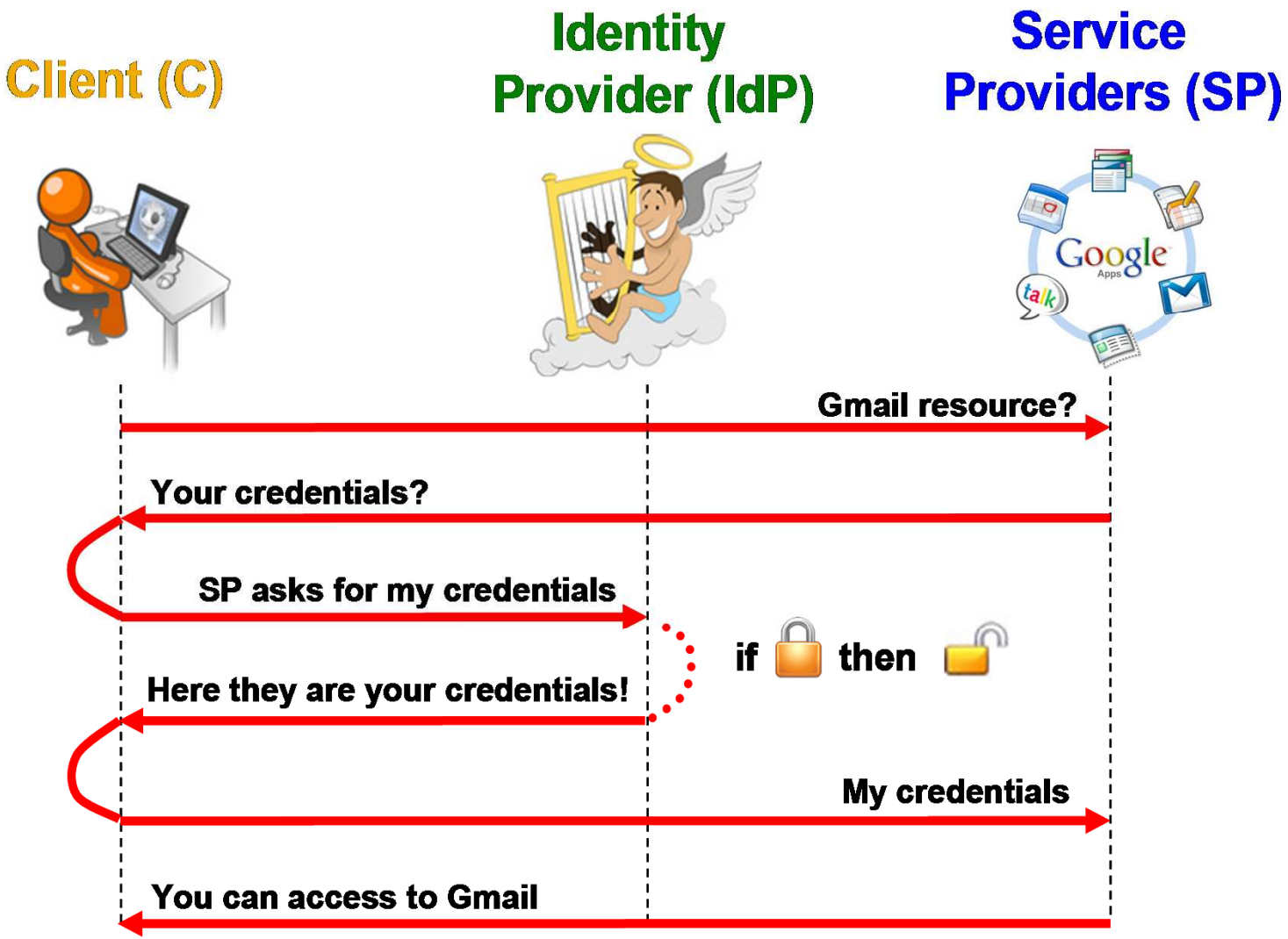
Scope of advancements wrt. state of the art



Technical Motivation: Google SAML-based Single Sign-On (SSO)



Motivation: SSO



SAML SSO

Deploying "secure" SAML-based services is not an easy task

1. SAML-based SSO for Google Apps (May 2008)

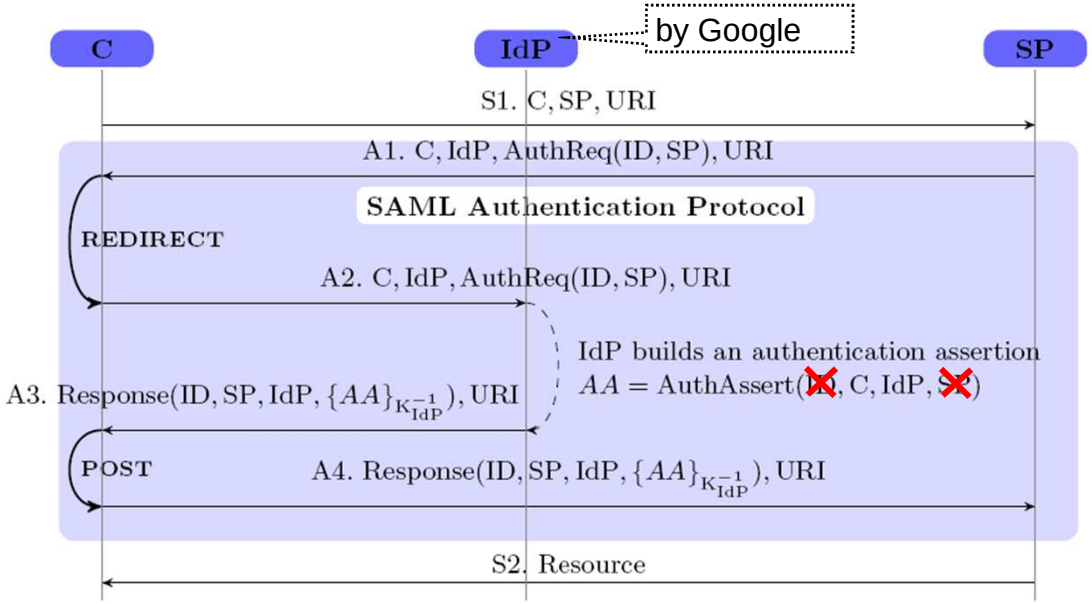


Fig. 1. SP-Initiated SSO with Redirect/POST Bindings

- ..a few but critical fields neglected in the IdP SSO service provisioned by Google..
- ..so that any SP can access to the Google's resources of IdP's members!

Abstract flaw automatically detected via automated reasoning (AVANTSSAR)

Attack manually tested on Google Apps. Is it possible to *automate?*

A worked-out example: SAML SSO

Deploying "secure" SAML-based services is not an easy task

1. SAML SSO authentication vulnerability and its exploitations (e.g., XSS in SAML-based SSO for Google Apps, July 2009)

   - C and SP interact twice. If no binding among these two interactions ⇨ vulnerability
   - vulnerability makes C consuming a resource from SP_2, while C asked for a resource from SP_1
   - serious or not serious? severity depends from how the abstract vulnerability can be exploited
     - serious for SAML-based SSO x Google Apps where the vulnerability could be exploited as launching pad for cross-Site Scripting (XSS): malicious SP able to get client cookies and unrestricted access to Google Apps under client's identity
     - not serious for other providers where the vulnerability could not be exploited like above

Abstract flaw automatically detected via automated reasoning (AVANTSSAR)

Consequent XSS discovered via human inspection (manual testing)

Can we *automate* this validation process?

A worked-out example: SAML SSO

What can we learn and what can we do?

- AVANTSSAR is excellent in discovering abstract service vulnerabilities on relevant deployment environments foreseen at design phase..
- ..but it is of little help in (i) assessing if an abstract vulnerability has serious exploitations in the real world, and (ii) detecting low-level pitfalls (e.g. XSS)

SPACIOS: combine automated reasoning with sophisticated testing techniques

SPaCIoS

A worked-out example: SAML SSO

Deployment environments and security requirements difficult to be foreseen when Google and OpenSSO are producing their own SAML services

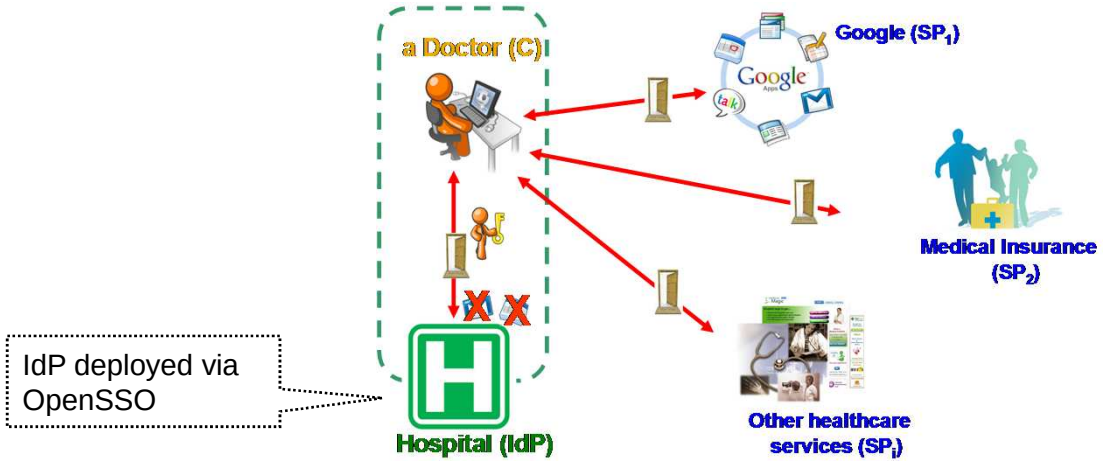
- A Hospital wants
  a) to outsource its basic IT services like email, calendar, etc to an external service provider that is offering specialized services in that area
- In outsourcing its basic IT services, the Hospital wants
  a) to keep the control of its identity management,
  b) to not add burden on its employees when they are using these services, and
  c) to have business continuity with its business partners (e.g., Medical Insurance)
- Aware that all its business partners (e.g., Medical Insurance) already offer a SAML SSO access, the Hospital decides:
  - to establish a SAML environment where it'll play as IdP to answer to both *b)*, *c)* and *d)*
  - to use OpenSSO to deploy the SAML IdP service on its machines
  - to use Google Apps for *a)*: Google Apps can be accessed via a SAML SSO

SPaCIoS

A worked-out example: SAML SSO

Deployment environments and security requirements difficult to be foreseen when Google and OpenSSO are producing their own SAML services (cont.)

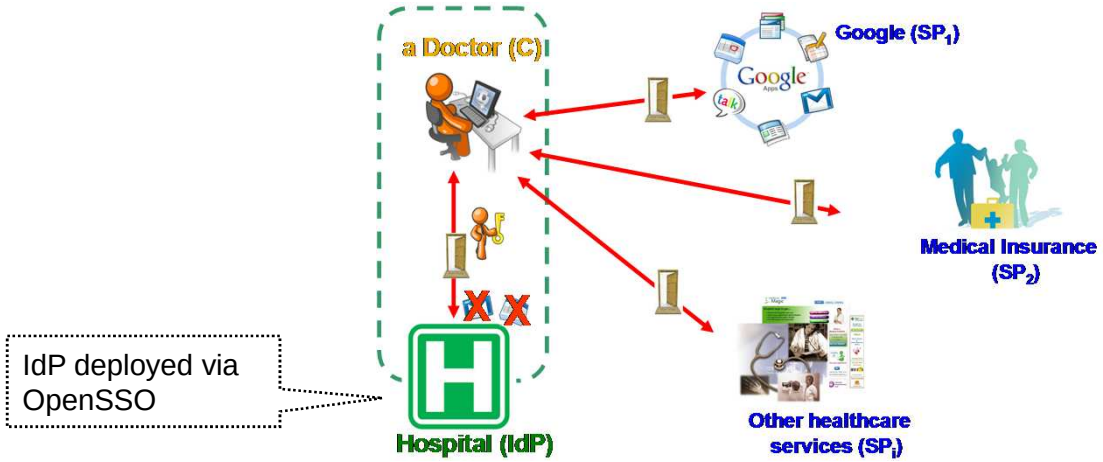
- The Hospital (H) just deploys IdP functionalities in its environment by means of OpenSSO
- Is the overall environment where H's IdP service is deployed secure wrt the H's requirements?
  - patient info must not be disclosed to unauthorized entities ⇨ e.g., Google email account of doctor X should be accessed by doctor X only



A worked-out example: SAML SSO

Deployment environments and security requirements difficult to be foreseen when Google and OpenSSO are producing their own SAML services (cont.)

- A new service may be deployed at runtime and made available for consumption
- Are consumers' security requirements met? E.g.,
  - assume Medical Insurance was providing a few services not dealing with sensible information and suddenly it deploys a new service dealing with consumer's sensible data
  - Clearly that new service will have to offer more security guarantees to the Hospital (i.e., Consumer)



A worked-out example: SAML SSO

What can we learn and what can we do?

- AVANTSSAR is excellent in discovering abstract service vulnerabilities on relevant deployment environments foreseen at design phase..
- ..but it is of little help in (i) assessing if an abstract vulnerability has serious exploitations in the real world, and (ii) detecting low-level pitfalls (e.g. XSS)

SPACIOS: combine automated reasoning with sophisticated testing techniques

- Not all deployment environments can be foreseen at design phase by Producers
- Intermediaries, Providers and even Consumers may bring in new security requirements

SPACIOS: validation has to be performed also at later stages (Deployment/Provision/Consumption)

SPaCIoS

Project objectives and approach

Problems:

- To achieve the needed guarantees to providers, intermediaries, and consumers of distributed services, rigorous security validation techniques must be applied.
- State-of-the-art security validation technologies fail to realise their full potential because they are typically used in isolation.
- Security validation in the Internet of Services (IoS) must be performed not only at production time, but also at deployment and consumption times.

Project objectives and approach:

- **Improve IoS security** by laying **technological foundations** of a new generation of security analysers for service deployment, provision and consumption.
- **Develop** the **SPaCIoS Tool** combining state-of-the-art technologies for penetration testing, model-based testing, model checking, and automatic learning.
- **Assess** the SPaCIoS Tool by running it against a set of security testing problem cases drawn from **industrial** and **open-source** IoS application **scenarios**.
- **Migrate** SPaCIoS technology to **industry** (SAP and Siemens business units), as well as to **standardisation** bodies and open-source communities.

SPaCIoS

The Consortium

Academia

- Università di Verona (I)
- ETH Zurich (CH)
- Grenoble INP (F)
- KIT Karlsruhe (D)
- Università di Genoa (I)

Industry

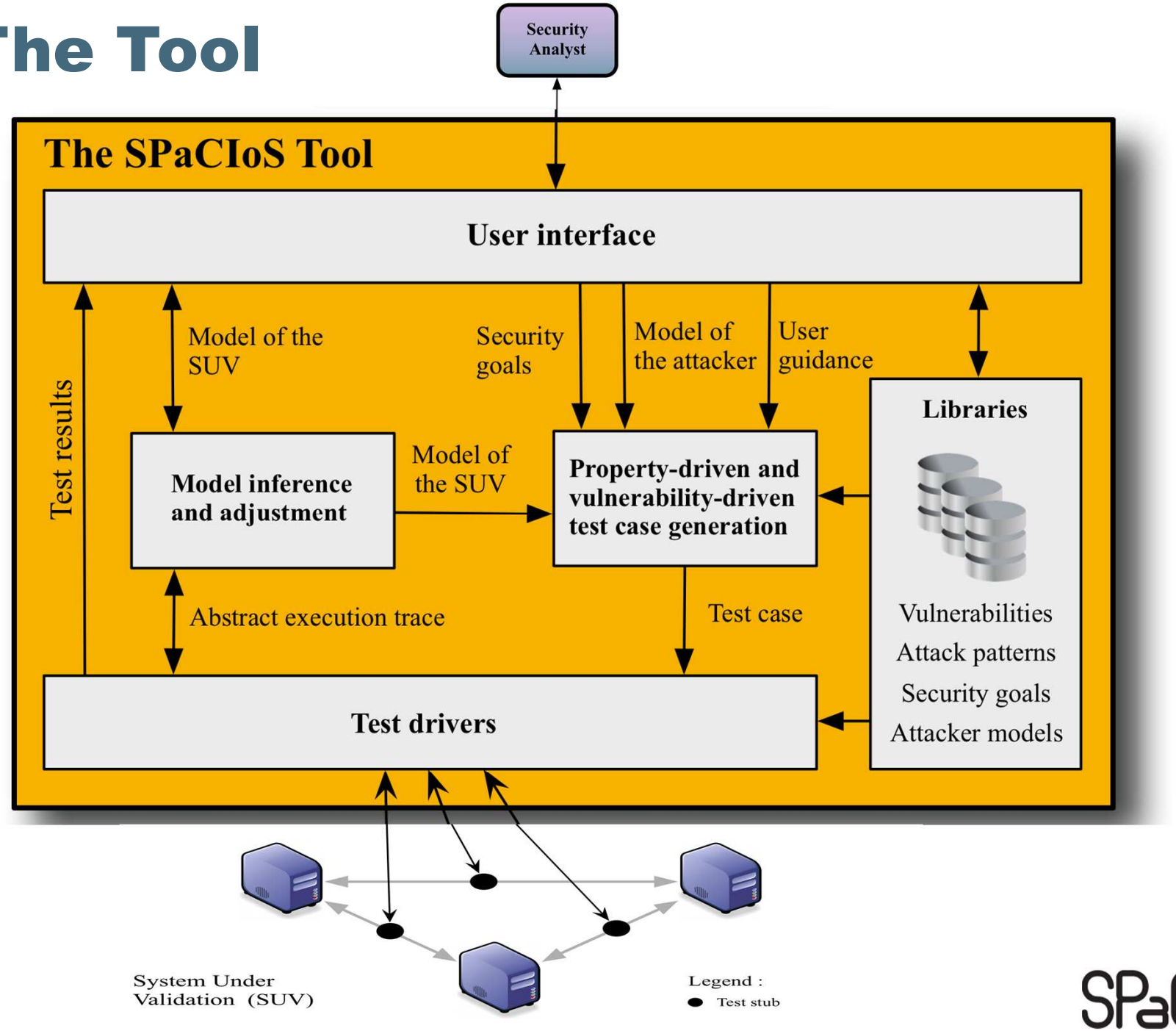
- SAP AG (D)
- Siemens AG Munich (D)

Expertise

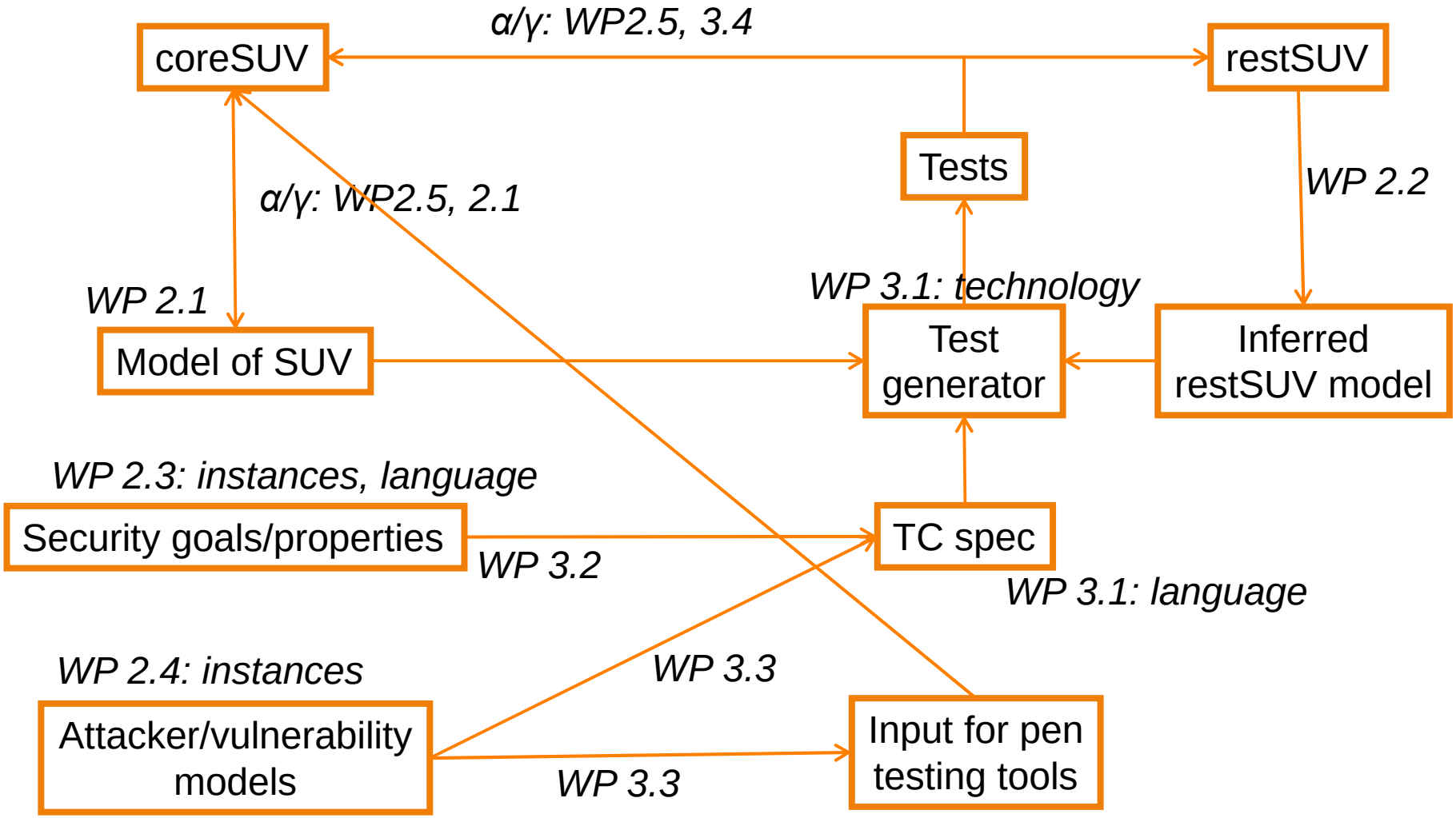
- Service-oriented architectures
- Security solutions
- Standardization and industry migration
- Automated security validation
- Formal methods and testing
- Security engineering

SPaCIoS

The Tool


Security Analyst
The SPaCIoS Tool
User interface
Model of the SUV
Security goals
Model of the attacker
User guidance
Test results
Model inference and adjustment
Model of the SUV
Property-driven and vulnerability-driven test case generation
Libraries
Abstract execution trace
Test case
Vulnerabilities
Attack patterns
Security goals
Attacker models
Test drivers
System Under Validation (SUV)
Legend :
Test stub

Technical core: WPs 2 and 3



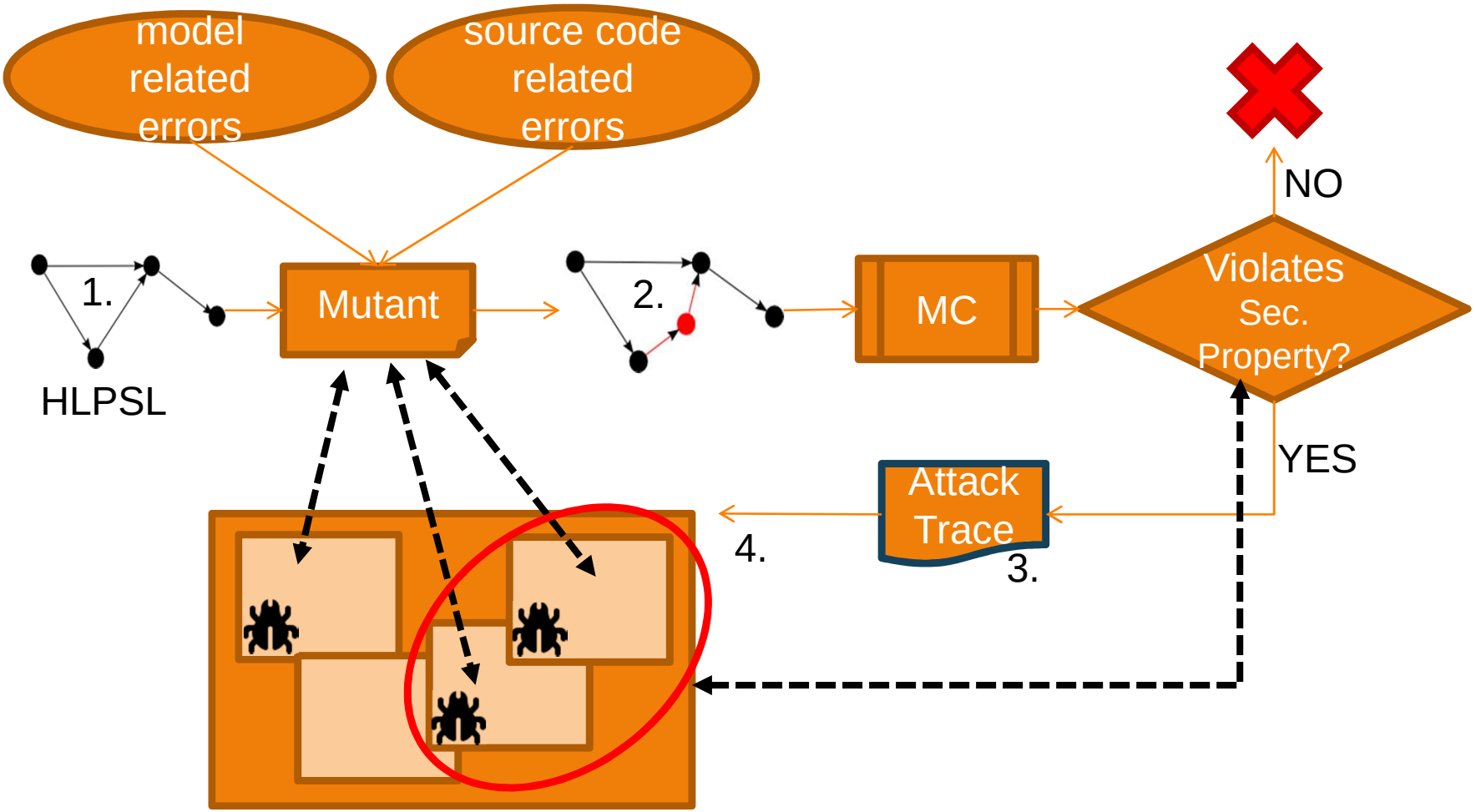
Scientific Core (KIT view)

- From models to systems
  - What abstractions do we apply, and what does this mean?
  - Bridge layers of abstraction between model and SUT (drivers)
- Property-based testing
  - Structural criteria don‘t correlate with failure detection. Period.
- Exploit model learning techniques
- Understand which technology is useful at which stages and where combinations are promising

- Is maybe modeling alone the key?

Current work at KIT

Security Mutants for Property Based Testing

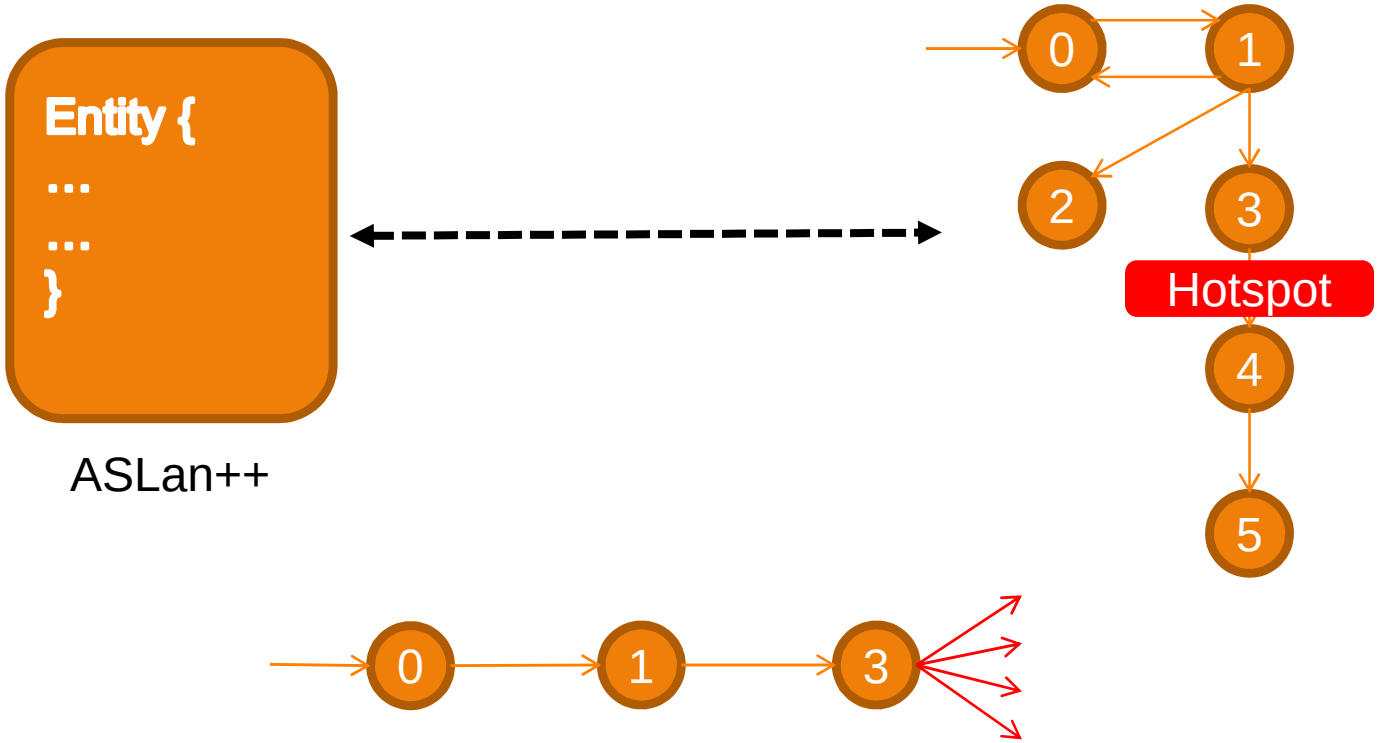


Current status of the mutant-based approach

- What we have done so far:
  - Applied to HLPSL: list of potential vulnerabilities and how to introduced them into the HLPSL model.
  - Note: A small number of attack traces generated.

- What we are currently working on:
  - Identify vulnerabilities than can be expressed in ASLan/ASLan++
  - Ex: In WebGoat lesson about stored XSS (cross site scripting), simple authentication implemented without using cookies.

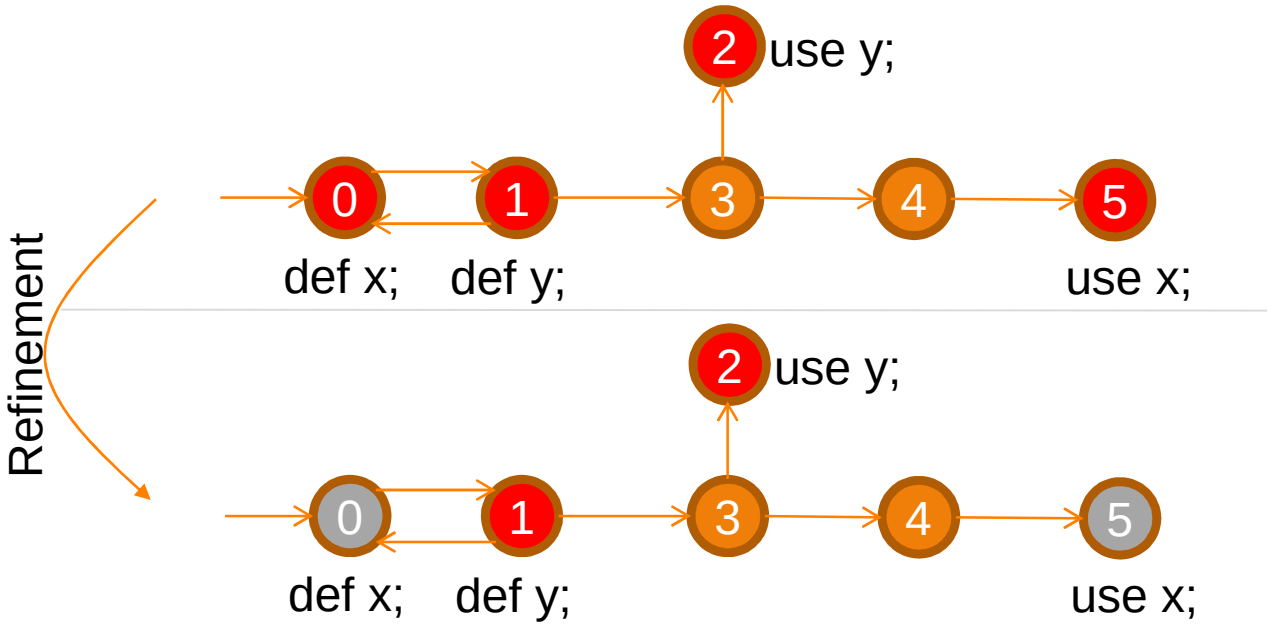
Hotspot: Concolic Testing For Security Vulnerabilities



1. Reach Hotspot
2. Try potential attacks from here

CEGAR for XSS Discovery

1. Assume non-sanitized inputs and outputs
2. Use model checker to find a trace that crosses an input and its corresponding output (def-use-coverage)
3. Try XSS attacks on this path
4. If no failure found, refine the model to mark this path as sanitized (no XSS possible on this path), and repeat steps 2 - 4



Wrap-Up

- Goal of SPACIOS:
  Bring the results from AVISPA and AVANTSSAR to the system level – „from models to systems“
- Abstractions conceptually and implementation-wise
- Understand how and where to combine tools