3rd UCAAT User Conference on Advanced Automated Testing

ETSI World Class Standards

Sophia Antipolis, French Riviera
20-22 October 2015

SECURITY THREAT IDENTIFICATION AND TESTING FOR SECURITY PROTOCOLS

Presented by Luca Compagna (SAP SE)

(joint work with Roberto Carbone, Annibale Panichella, Serena Ponta)

STIATE

SAP FONDAZIONE BRUNO KESSLER eit Digital

Context: Multi-Party Web Applications

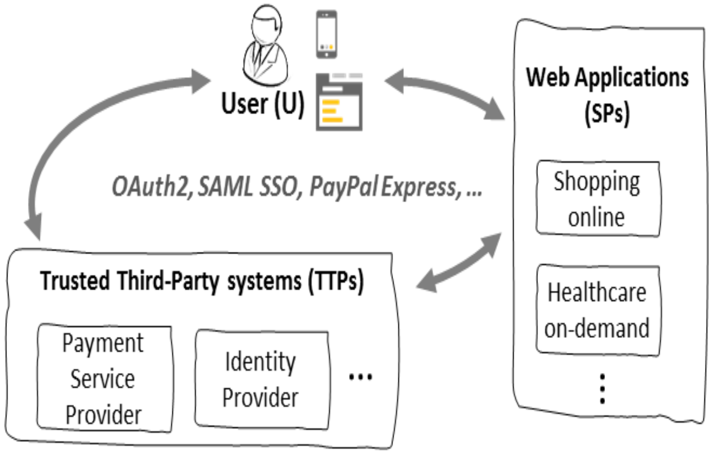
Many modern web applications relies on TTPs to deliver services to their Users

- e.g., 27% of Alexa top 1000 uses Facebook SSO

Based on:

- protocols (interoperability)
- bilateral trust relationships

TTPs are assumed to be trustworthy
But neither SP nor C are assumed so



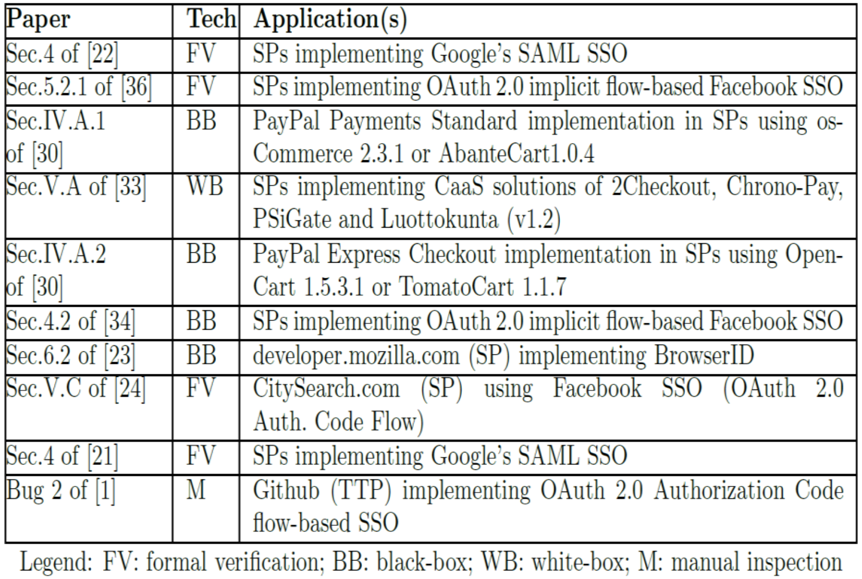
Challenges and Motivations

Several vulnerabilities reported in literature

Mainly **implementation** issues, but also **design** ones

Challenges include:

- highly **configurable** protocols, **interpretation** of the specifications
- **internal requirements**, total cost for development (**TCD**)
    - lack of (security) **testing**, but also
    - lack of **tool support** for developers
- …



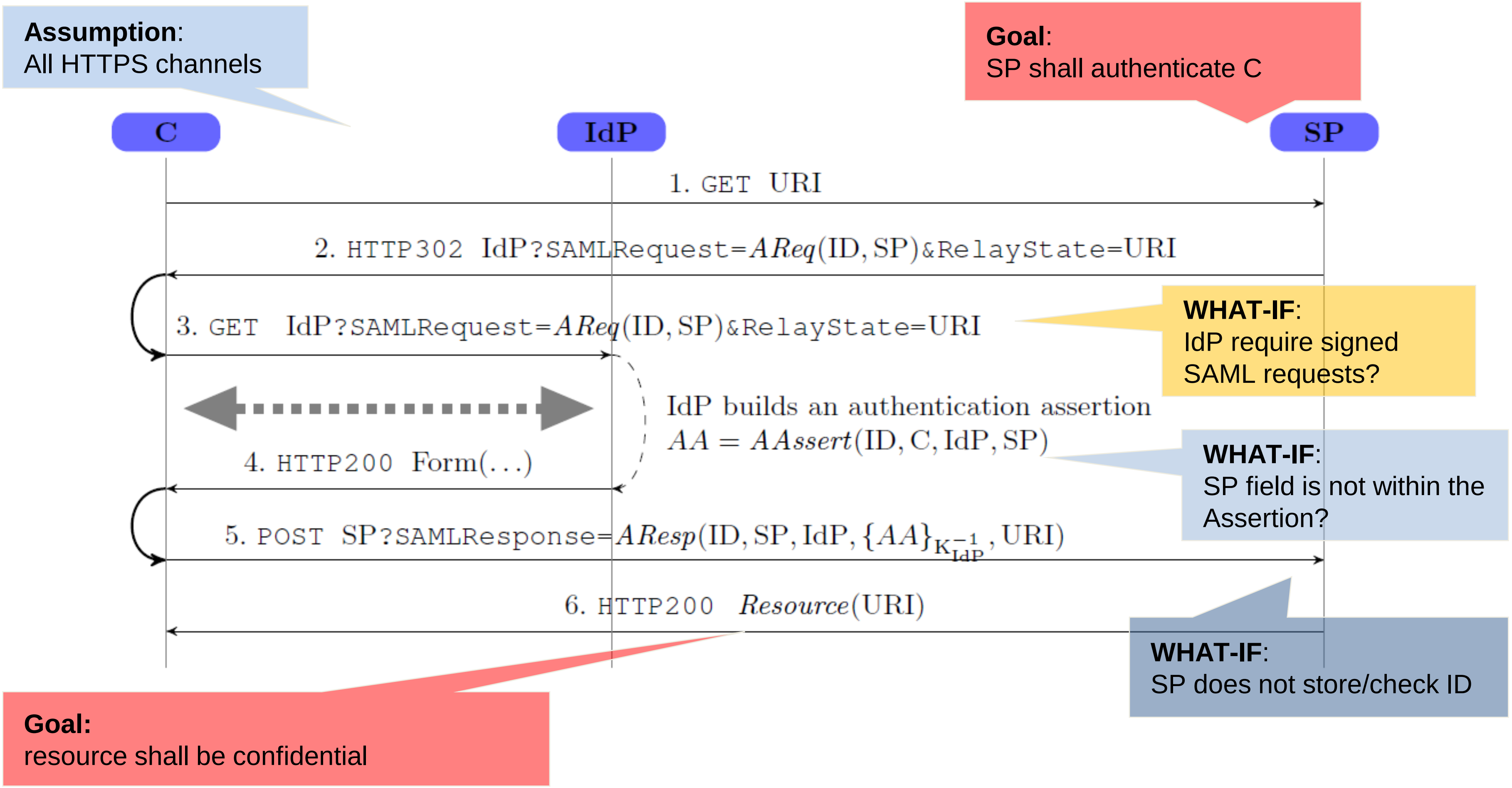
Paper	Tech	Application(s)
Sec.4 of [22]	FV	SPs implementing Google's SAML SSO
Sec.5.2.1 of [36]	FV	SPs implementing OAuth 2.0 implicit flow-based Facebook SSO
Sec.IV.A.1 of [30]	BB	PayPal Payments Standard implementation in SPs using os-Commerce 2.3.1 or AbanteCart1.0.4
Sec.V.A of [33]	WB	SPs implementing CaaS solutions of 2Checkout, Chrono-Pay, PSiGate and Luottokunta (v1.2)
Sec.IV.A.2 of [30]	BB	PayPal Express Checkout implementation in SPs using Open-Cart 1.5.3.1 or TomatoCart 1.1.7
Sec.4.2 of [34]	BB	SPs implementing OAuth 2.0 implicit flow-based Facebook SSO
Sec.6.2 of [23]	BB	developer.mozilla.com (SP) implementing BrowserID
Sec.V.C of [24]	FV	CitySearch.com (SP) using Facebook SSO (OAuth 2.0 Auth. Code Flow)
Sec.4 of [21]	FV	SPs implementing Google's SAML SSO
Bug 2 of [1]	M	Github (TTP) implementing OAuth 2.0 Authorization Code flow-based SSO

Legend: FV: formal verification; BB: black-box; WB: white-box; M: manual inspection

[1] Account hijacking by leaking authorization code. http://www.oauthsecurity.com/.
[21] Armando, A., Carbone, R., Compagna, L., Cuellar, J., Pellegrino, G., and Sorniotti, A. From multiple credentials to browser-based single sign-on: Are we more secure? IFIP 2011.
[22] Armando, A., Carbone, R., Compagna, L., Cuellar, J., and Tobarra, L. Formal Analysis of SAML 2.0 Web Browser Single Sign-On: Breaking the SAML-based Single Sign-On for Google Apps. FMSE 2008
[24] Bai, G., Lei, J., Meng, G., Venkatraman, S. S., Saxena, P., Sun, J., Liu, Y., and Dong, J. S. Authscan: Automatic extraction of web authentication protocols from implementations. NDSS 2013
[30] Pellegrino, G., and Balzarotti, D. Toward black-box detection of logic flaws in web applications. NDSS 2014
[33] Sun, F., Xu, L., and Su, Z. Detecting logic vulnerabilities in e-commerce applications. NDSS 2014
[34] Wang, R., Chen, S., and Wang, X. Signing me onto your accounts through facebook and google: A traffic-guided security study of commercially deployed single-sign-on web services. S&P 2012
[36] Wang, R., Zhou, Y., Chen, S., Qadeer, S., Evans, D., and Gurevich, Y. Explicating SDKs: Uncovering assumptions underlying secure authentication and authorization. USENIX 2013

Illustrative example

Developing and deploying SAML SSO



Assumption:
All HTTPS channels

Goal:
SP shall authenticate C

C

IdP

SP

1. GET URI

2. HTTP302 IdP?SAMLRequest=$AReq$(ID, SP)&RelayState=URI

3. GET IdP?SAMLRequest=$AReq$(ID, SP)&RelayState=URI

WHAT-IF:
IdP require signed SAML requests?

IdP builds an authentication assertion
$AA = AAssert(\mathrm{ID}, \mathrm{C}, \mathrm{IdP}, \mathrm{SP})$

WHAT-IF:
SP field is not within the Assertion?

4. HTTP200 Form(...)

5. POST SP?SAMLResponse=$AResp(\mathrm{ID}, \mathrm{SP}, \mathrm{IdP}, \{AA\}_{K_{\mathrm{IdP}}^{-1}}, \mathrm{URI})$

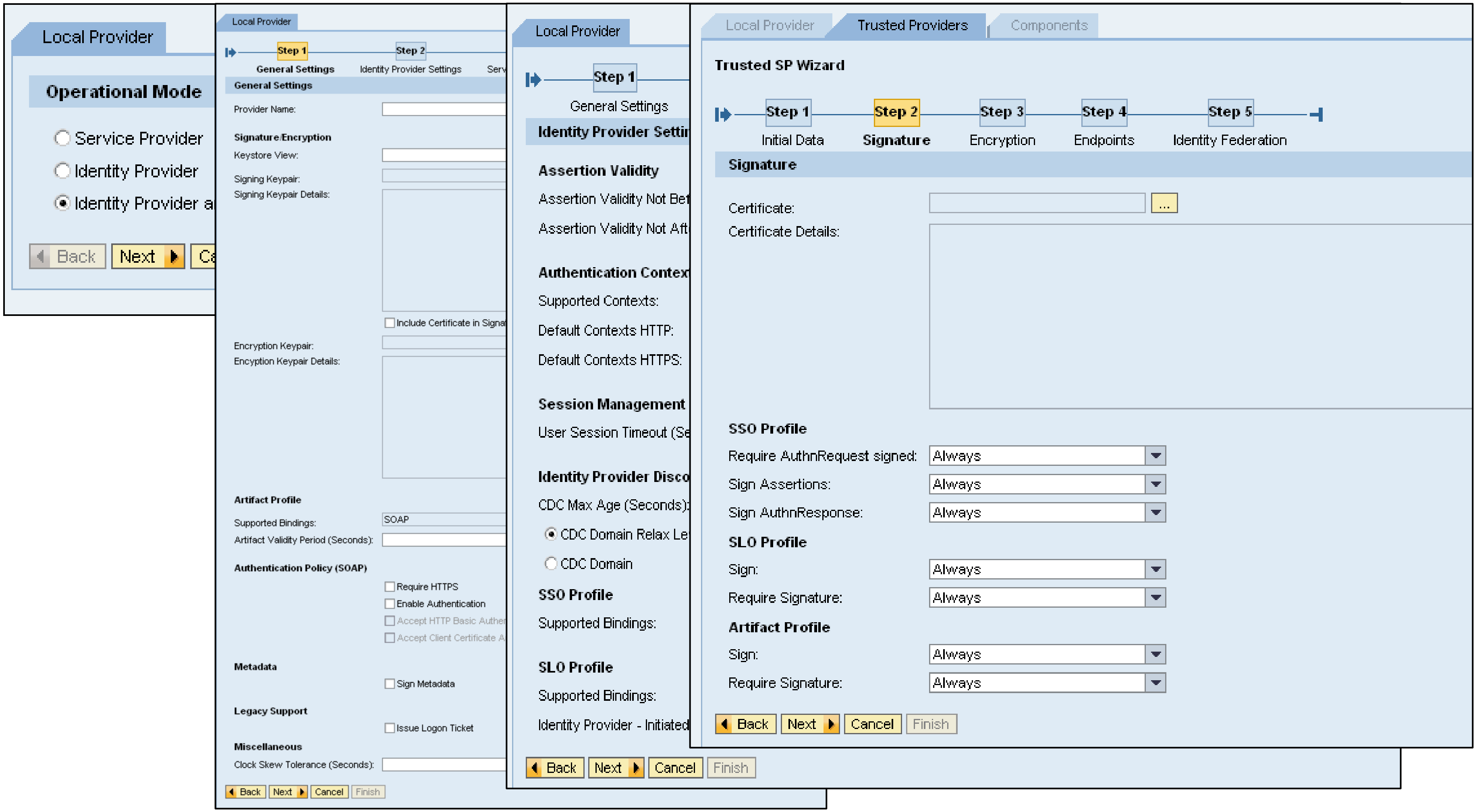
6. HTTP200 *Resource*(URI)

WHAT-IF:
SP does not store/check ID

Goal:
resource shall be confidential

SAML2 comes with many profiles, protocols, optional attributes, etc... **+** Internal requirements
= several **WHAT-IF**

Illustrative example
Developing and deploying SAML SSO



Local Provider

Operational Mode

○ Service Provider

○ Identity Provider

● Identity Provider a

Back | Next | Ca

Local Provider

Step 1 General Settings | Step 2 Identity Provider Settings | Serv

General Settings

Provider Name:

Signature/Encryption

Keystore View:

Signing Keypair:

Signing Keypair Details:

□ Include Certificate in Signa

Encryption Keypair:

Encyption Keypair Details:

Artifact Profile

Supported Bindings: SOAP

Artifact Validity Period (Seconds):

Authentication Policy (SOAP)

□ Require HTTPS

□ Enable Authentication

□ Accept HTTP Basic Authen

□ Accept Client Certificate A

Metadata

□ Sign Metadata

Legacy Support

□ Issue Logon Ticket

Miscellaneous

Clock Skew Tolerance (Seconds):

Back | Next | Cancel | Finish

Local Provider

Step 1 General Settings

Identity Provider Settir

Assertion Validity

Assertion Validity Not Bef

Assertion Validity Not Aft

Authentication Contex

Supported Contexts:

Default Contexts HTTP:

Default Contexts HTTPS:

Session Management

User Session Timeout (Se

Identity Provider Disco

CDC Max Age (Seconds):

● CDC Domain Relax Le

○ CDC Domain

SSO Profile

Supported Bindings:

SLO Profile

Supported Bindings:

Identity Provider - Initiated

Back | Next | Cancel | Finish

Local Provider | Trusted Providers | Components

Trusted SP Wizard

Step 1 Initial Data | Step 2 Signature | Step 3 Encryption | Step 4 Endpoints | Step 5 Identity Federation

Signature

Certificate: ...

Certificate Details:

SSO Profile

Require AuthnRequest signed: Always

Sign Assertions: Always

Sign AuthnResponse: Always

SLO Profile

Sign: Always

Require Signature: Always

Artifact Profile

Sign: Always

Require Signature: Always

Back | Next | Cancel | Finish

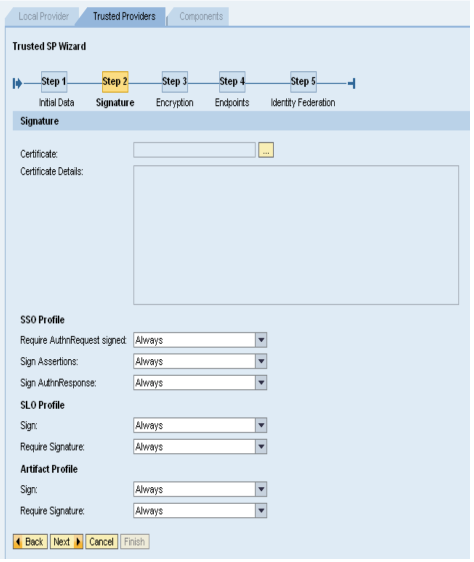
source: few screen-shots of the SAP NetWeaver SAML Next Generation Single Sign On

Illustrative example

Developing and deploying SAML SSO

#	from	SSL/TLS				Signature			Encryptions	C checks on AuthnReq			SP checks ID
		C-IdP	C-SP (1)	C-SP (2)	(1) = (2)	Authn Req	Assertions	Authn Resp		IdP	SP	URI	

Properties		
authentication	resource secrecy	Stealing C's authn material



Trusted SP Wizard

Signature

Certificate:

Certificate Details:

SSO Profile

Require AuthnRequest signed:	Always
Sign Assertions:	Always
Sign AuthnResponse:	Always

SLO Profile

Sign:	Always
Require Signature:	Always

Artifact Profile

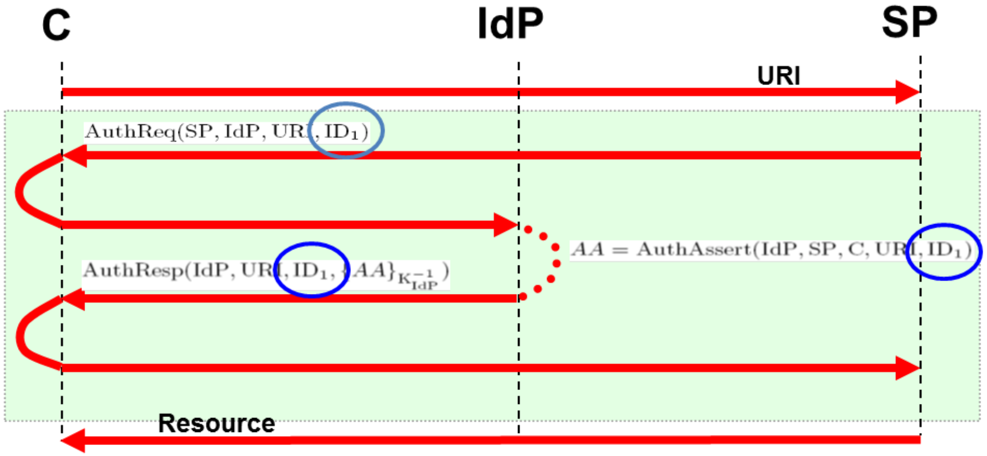
Sign:	Always
Require Signature:	Always

Purpose: identify SAFE vs UNSAFE configurations in the WHAT-IF space

Illustrative example

Developing and deploying SAML SSO

		SSL/TLS				Signature				C checks on AuthnReq				Properties		
#	from	C-IdP	C-SP (1)	C-SP (2)	(1) = (2)	Authn Req	Assertions	Authn Resp	Encryptions	IdP	SP	URI	SP checks ID	authentication	resource secrecy	Stealing C's authn material


C
IdP
SP
URI
$\mathrm{AuthReq}(\mathrm{SP}, \mathrm{IdP}, \mathrm{URI}, \mathrm{ID}_1)$
$AA = \mathrm{AuthAssert}(\mathrm{IdP}, \mathrm{SP}, \mathrm{C}, \mathrm{URI}, \mathrm{ID}_1)$
$\mathrm{AuthResp}(\mathrm{IdP}, \mathrm{URI}, \mathrm{ID}_1, \{AA\}_{K^{-1}_{\mathrm{IdP}}})$
Resource

Purpose: identify SAFE vs UNSAFE configurations in the WHAT-IF space

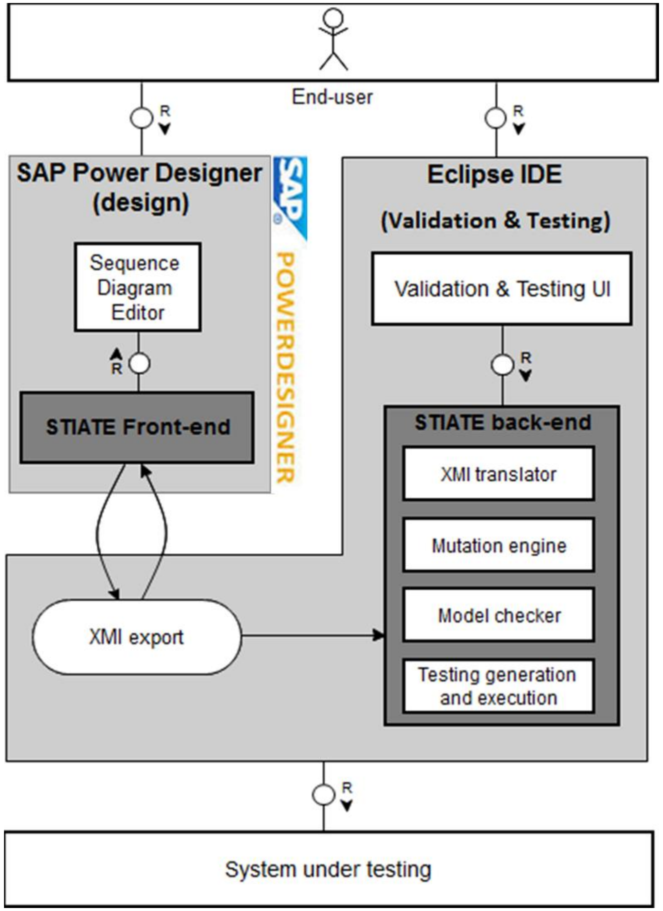
Our solution

identify **SAFE vs UNSAFE** configurations in the **WHAT-IF** space

- Threat identification at design-time via model-checking
- Model-based testing

rigorous, but **viable** for an **industrial setting**

- accessibility / usability
- automation / integration
- cost-benefit ratio (TCO)



Our solution (cont.)

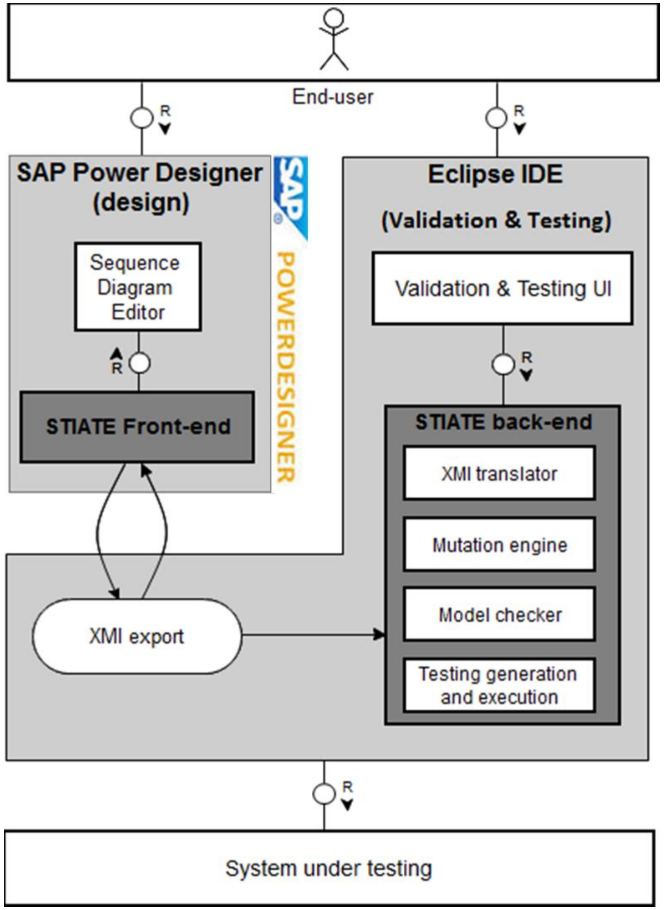
Sequence Diagram
(informal model)

↓

Front-end
add-in for Enterprise Modelling Tools (SAP Power Designer)

↓

Security Annotations
(XMI files)



XMI Translator
converts XMI files into Formal Models

↓

Model Checker
SATMC performs the formal analysis

↓

Mutations
mutate the formal models

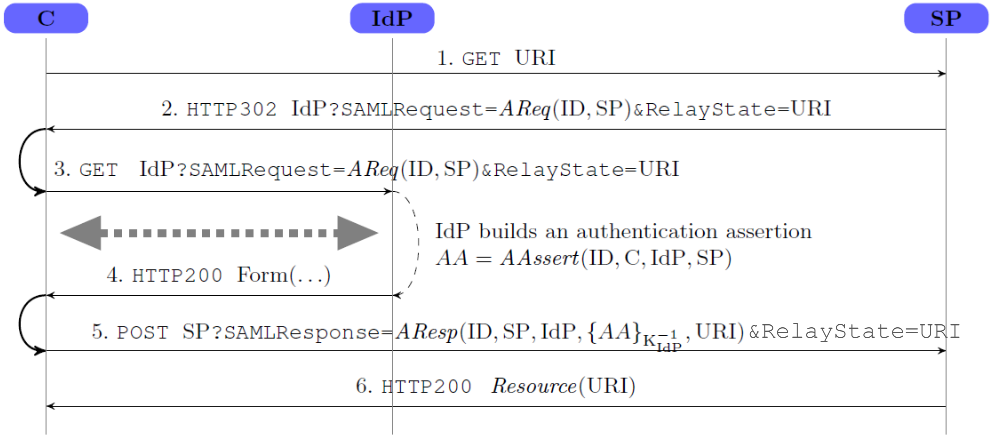
↓

Test Case Execution
(model-based testing)

Scenario: SAML SSO

SAML 2.0 Web Browser SSO Profile:

- SAML-based SSO for Google Apps
- Novell Access Manager
- SimpleSAMLphp by UNINETT



1. GET URI

2. HTTP302 IdP?SAMLRequest=$AReq(\mathrm{ID}, \mathrm{SP})$&RelayState=URI

3. GET IdP?SAMLRequest=$AReq(\mathrm{ID}, \mathrm{SP})$&RelayState=URI

IdP builds an authentication assertion
$AA = AAssert(\mathrm{ID}, \mathrm{C}, \mathrm{IdP}, \mathrm{SP})$

4. HTTP200 Form(...)

5. POST SP?SAMLResponse=$AResp(\mathrm{ID}, \mathrm{SP}, \mathrm{IdP}, \{AA\}_{\mathrm{K}_{\mathrm{IdP}}^{-1}}, \mathrm{URI})$&RelayState=URI

6. HTTP200 $Resource(\mathrm{URI})$

Scenario: SAML SSO (demo)

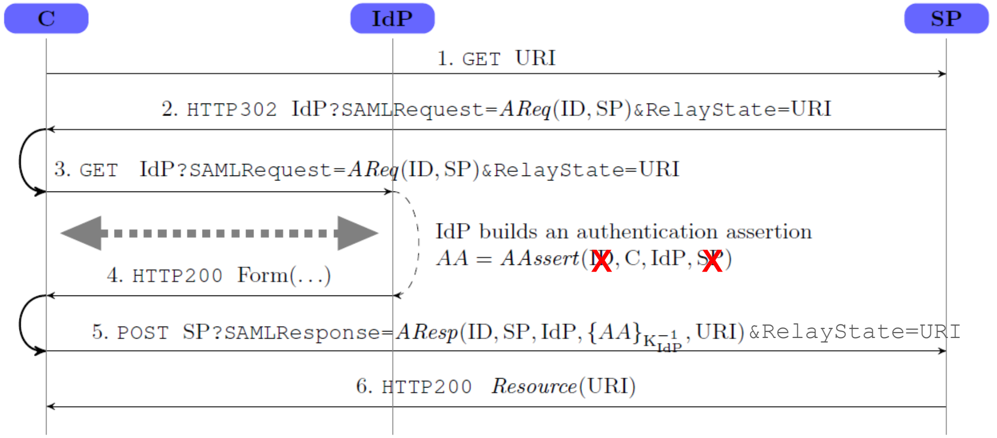
SAML 2.0 Web Browser SSO Profile:

- SAML-based SSO for Google Apps
- Novell Access Manager
- SimpleSAMLphp by UNINETT

Vulnerabilities due to wrong design choices

(Armando et al. Formal Analysis of SAML 2.0 Web Browser Single Sign-On: Breaking the SAML-based Single Sign-On for Google Apps. FMSE 2008)

Man-in-the-middle **attack** due to missing fields **SP** and **ID** in the **assertion**



C IdP SP

1. GET URI

2. HTTP302 IdP?SAMLRequest=$AReq(\mathrm{ID}, \mathrm{SP})$&RelayState=URI

3. GET IdP?SAMLRequest=$AReq(\mathrm{ID}, \mathrm{SP})$&RelayState=URI

IdP builds an authentication assertion
$AA = AAssert(\cancel{\mathrm{ID}}, \mathrm{C}, \mathrm{IdP}, \cancel{\mathrm{SP}})$

4. HTTP200 Form(...)

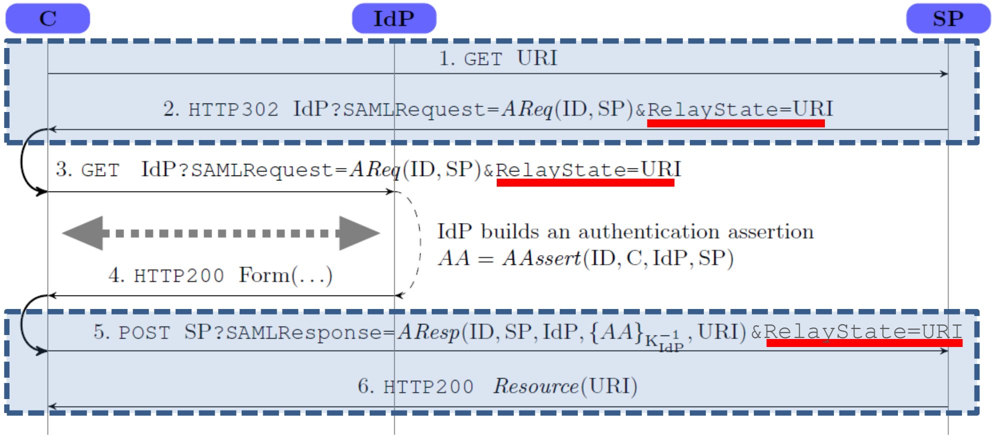
5. POST SP?SAMLResponse=$AResp(\mathrm{ID}, \mathrm{SP}, \mathrm{IdP}, \{AA\}_{\mathrm{K}_{\mathrm{IdP}}^{-1}}, \mathrm{URI})$&RelayState=URI

6. HTTP200 $Resource(\mathrm{URI})$

Scenario: SAML SSO (demo)

SAML 2.0 Web Browser SSO Profile:

- SAML-based SSO for Google Apps
- Novell Access Manager
- SimpleSAMLphp by UNINETT



C

IdP

SP

1. GET URI

2. HTTP302 IdP?SAMLRequest=$AReq(\mathrm{ID}, \mathrm{SP})$&RelayState=URI

3. GET IdP?SAMLRequest=$AReq(\mathrm{ID}, \mathrm{SP})$&RelayState=URI

IdP builds an authentication assertion
$AA = AAssert(\mathrm{ID}, \mathrm{C}, \mathrm{IdP}, \mathrm{SP})$

4. HTTP200 Form(...)

5. POST SP?SAMLResponse=$AResp(\mathrm{ID}, \mathrm{SP}, \mathrm{IdP}, \{AA\}_{K^{-1}_{\mathrm{IdP}}}, \mathrm{URI})$&RelayState=URI

6. HTTP200 $Resource(\mathrm{URI})$

Vulnerabilities due to wrong design choices

(Armando et al. Formal Analysis of SAML 2.0 Web Browser Single Sign-On: Breaking the SAML-based Single Sign-On for Google Apps. FMSE 2008)

Man-in-the-middle **attack** due to missing fields **SP** and **ID** in the **assertion**

Design vs. Implementation

(Armando et al. An authentication flaw in browser-based Single Sign-On protocols: Impact and remediations. Computers & Security 2013)

XSS **attack** due to unrealistic **assumption** and missing **input validation**

Final remarks

Proof-of-concept READY

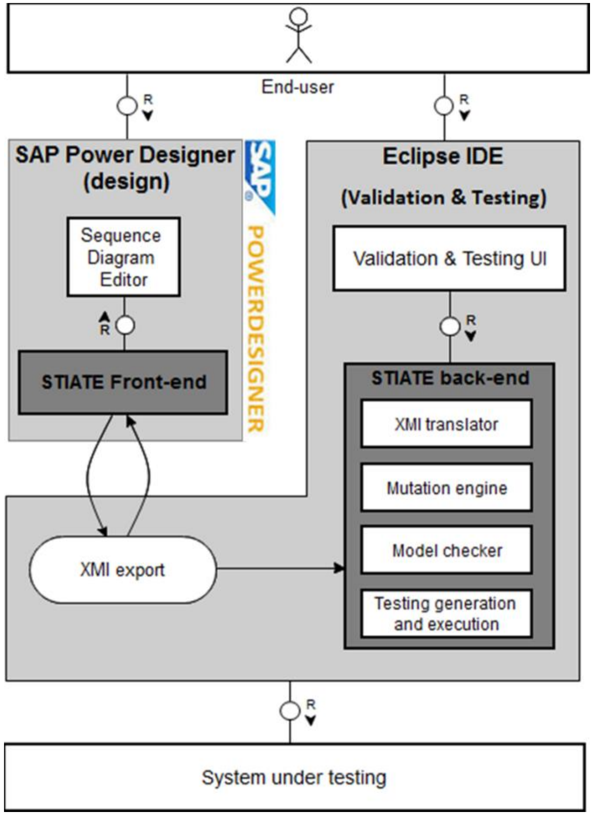
- prototype integrated within SAP Power Designer
- other use cases under scrutiny: e.g., mobile payment commercial solution

Potential end-users

- Architects and development teams integrating a core security protocol
- Security consultants analyzing a customer proprietary protocol (e-payment)
- Standardization bodies designing protocols and reference implementations

Industrial transfer (our experience)

- though lowered, the TCD is still not negligible
- consultancy mode works well, handing over the prototype not so well



THANK YOU

Contact: luca.compagna@sap.com