SAML Profile Test Framework

Author: Rainer Hörbe (City of Vienna)
Date: 10-Feb-2012
Document Status: Initial draft (0.2)

Scope and Purpose

Assure compliance of federation-facing interfaces of SAML actors

The SAML Profile Test Framework (referred here „Framework“) shall provide (prospective) participants of federations a tool to assess their products and services for interoperability and compliance with a specific SAML profile. It shall confine the SAML WebSSO use case to the specifications and requirements of a specific deployment. A list of deployment profiles related to the Kantara SAML 2.0 eGov Interoperability Profiles can be found at the Kantara FI-WG Wiki[1] .

The Framework is designed as a community service that shall be provided as a kind of cloud service with a common repository of test cases. Tests can use and contribute to the repository thus resembling a crowd-sourced approach to testing.

The focus in the first phase is to provide test services to SPs. The rationale is for portalverbund.at[2] that there is a significantly lager number of SPs than IdPs, and IdPs usually have more experience with IAM than SPs.

Stakeholders

Stakeholders addressed by this effort are:

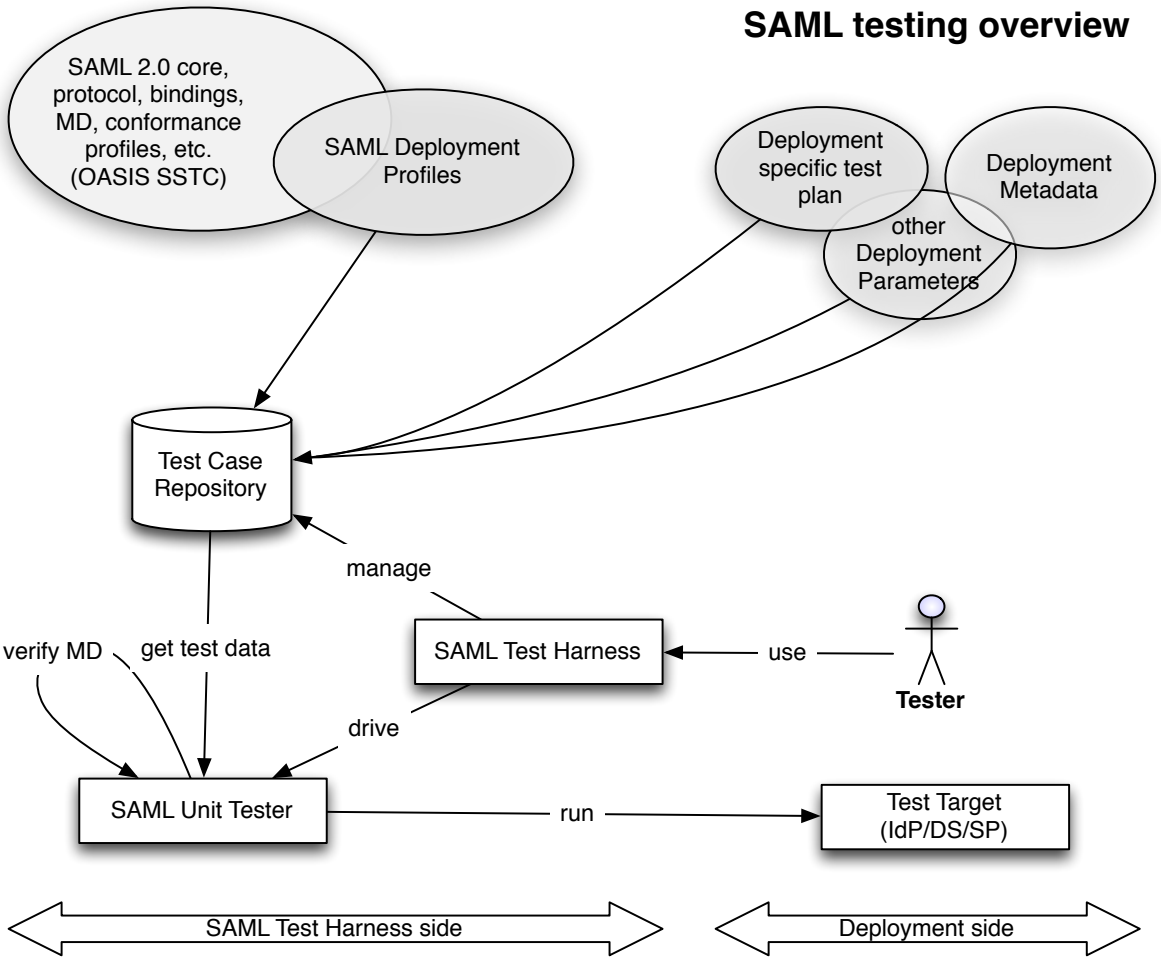
Stakeholder	*Concern*
SAML Profile Owner	Ensure that profile is adhered to; collect feedback from testers to improve profile.
Federation Authority/Operator	Require entities to be profile compliant before new or updates software is deployed.
Product vendor, OSS supporter/contributor	Ensure product compliance with deployment requirements; Reduce support requests by using test suite to spot configuration errors.
IdP Operator	Reduce integration effort with SPs
SP Operator	Reduce integration efforts when connecting to federations
IDM Initiatives (REFEDS, Kantara, Géant)	Improved SAML interoperability and adoption of federation technology

The drivers of this effort are Kantara Initiative, Géant, and the AG-IZ (Austrian government’s eGov federation WG).

[1] http://kantarainitiative.org/confluence/display/fiwg/SAML+Interoperability+and+Deployment+Profiles
[2] Austrian G2G Federation

The first SAML profile to be implemented will be PVP2-S-Profile V2.1 (www.portalverbund.at)

Components



Test case repository ("Repo")

It stores the test data that is used by the *SAML Unit Tester* to execute tests.

- SAML standards (from OASIS SSTC);
- SAML Deployment Profile (restricts and extends the SAML standards);
- Deployment metadata (provides specific values about the SAML actors);
- Other deployment parameters: data and decisions that are neither in the profile nor in MD, but are needed to provide complete configurations; e.g. attribute sets and values.
- Unit test cases that prove certain behavior and structures.
- Management information enabling *Testers* to manage test scenarios, results and access rights.

The Repo shall be a common resource for many deployments. Test cases should be shared and improved in a community effort, but SAML profiles, test cases, test federations and test execution may differ for specific deployments. To support such a model, a service-type test case repository is being proposed as a single instance for all interested stakeholders.

Ideally, the repository should contain a superset of all test cases for all deployments to be tested providing reasonable test coverage. Test cases can be grouped and parameterized to address certain test scenarios. An extension and inheritance schema could be used to organize multiple scenarios.

Test execution would be the task of deployment-specific infrastructure. The tester would configure the test data and execute the tests. There should be no need to install local instances of the test framework.

SAML Test Harness

This is a GUI application to manage the repo, invoke and analyze tests. Besides the traditional functions of a test harness it shall manage:

- parameterization of generic unit tests with site specific parameters
- dependencies between operations (probably relying on python inheritance mechanism)
- useful categorization of test data that allows testers to quickly set up comprehensive tests (although it is still a human effort to verify what comprehensive actually means. Automated test coverage analysis will probably be too complex.)

SAML Unit Tester

Invoked by the test harness for a specific unit test in a specified context, it emulates a communication partner to the *Test Target*. E.g. if the test target is an IdP, the unit tester emulates a web browser with a user and a SP.
Each unit test (internally called "operation") comprises 3 groups:

1. Pre-interaction check
2. Interaction
3. Post-interaction check

Each group is implemented as a sequence of Python classes that use pysaml2.
Sample unit tests that are currently implemented are:

- AuthnRequest using HTTP POST expecting transient NameID
- AuthnRequest and then an AuthnQuery
- SAML2 AuthnRequest using ECP and PAOS
- AuthnRequest using HTTP-redirect followed by a logout

Test Categories

SAML actors to be tested are SP and IdP.

Metadata Correctness & Completeness
- Schema valid XML?
- Check on elements in unknown namespaces
- Warn on recommended but missing elements
- Certificate validity
- Endpoint availability

Protocol flow
- Support for different bindings
- Request formats (control of AuthnRequest elements)
- Response contents (attribute sets, attribute values and rules involving multiple attributes)

- LoA including timeout compliance

Crypto properties
- Cipher support
- Signatures & TLS where required by profile
- trust anchors
- error handling on invalid, expired signatures and TLS-certs

Subject Attribute test
- Nameid
- Attribute

Other rules
- retain relay state between request/response
- execute access decision based on authContextClassRef (exact match)
- Metadata freshness rules obeyed?

Vulnerability Scan
- Signature und TLS cert validation
- XML-Signature Wrapping
- IMHO tests shoul include HTTP-server vulnerability scans like the OWASP 10