Testing for SHA-2 Compatibility

Testing for SHA-2 Compatibility in the InCommon Federation

Tom Barton and Tom Scavo identified the following steps that a tester would take:

1. Preliminary conference call to go over the details, make any adjustments.
2. Testers deploy **two** test IdPs (see Phase 2 section below for details).
3. Add the test IdPs to InCommon metadata.
4. Set up or identify an account to be used for testing---this is the principal for which an authentication assertion will be sent to each InCommon SP. The account should be configured for HTTP Basic Authentication.
5. Coordinate with TomS who will run an HTTP client to send unsolicited SAML2 Response to each InCommon SP.
    - The client code will run remotely, from an unspecified location.
    - All logging will be done at the client.
    - The tester will help troubleshoot the SAML2 exchange if necessary.
6. TomS will gather results from all test instances on a wiki page for all of us to see.
7. TomS, Nate Klingenstein, TomB, and others will review the results, contact some SP operators, and investigate apparent issues. Testers might need to be looped in to help get to the bottom of some issues - we'll see.
8. Wrap up conference call to review the results after the SP analysis phase is done.

HTTP Client

Leveraging the IdP-initiated SSO feature of the Shibboleth IdP, an HTTP client will authenticate to the IdP via HTTP Basic Authentication and push a signed SAML2 Response to a SAML2 HTTP-POST endpoint at the SP. The client code will be a fork of the dead entity scripts, a set of bash scripts that probe endpoints in InCommon metadata. (Some sample code is provided so you can see where we're headed.)

At the end of this exercise, the HTTP client code will be released as open source.

Test IdPs

Since the Shibboleth IdP supports only one signature/digest algorithm at a time, two separate Shibboleth IdP deployments are required.

IdP #1

- A standard Shibboleth IdP deployment (IdP #1) that supports:
    - SAML2 Web Browser SSO (SAML1 is not required)
    - IdP-initiated SSO
    - HTTP Basic Authentication
    - SHA-1 XML Signatures (the Shib IdP default)
- The SAML2 Response element is signed with a SHA-1 XML signature
- The SAML2 Assertion element is unsigned and unencrypted
- The metadata for IdP #1 must be registered in the InCommon metadata aggregate
- The Attribute Release Policy for IdP #1 must release a Transient ID to any InCommon SP

IdP #2

- A custom Shibboleth IdP deployment (IdP #2) that supports:
    - SAML2 Web Browser SSO (SAML1 is not required)
    - IdP-initiated SSO
    - HTTP Basic Authentication
    - SHA-256 XML Signatures (via a Shib IdP extension)
- The SAML2 Response element is signed with a SHA-256 XML signature
- The SAML2 Assertion element is unsigned and unencrypted
- The metadata for IdP #2 must be registered in the InCommon metadata aggregate
- The Attribute Release Policy for IdP #2 must release a Transient ID to any InCommon SP

The only difference between the two IdPs is the signature/digest algorithm used to sign the SAML2 Response.

Testing Strategy

Phase 1 - TomS finalizes HTTP Client code

At least one tester's IdP infrastructure is required for this phase of the work.

For each SP in InCommon metadata, the HTTP Client does the following:

1. If the SP does not support SAML2 HTTP-POST, skip to the next SP
2. Probe the SP for liveness by sending an empty POST request to the SP's SAML2 HTTP-POST endpoint
3. If the SP does not respond, skip to the next SP
4. Push a SHA-1 signed SAML2 Response from IdP #1 to the SP's SAML2 HTTP-POST endpoint
5. Continue with the next SP

Phase 2 - Gather SHA-256 compatibility data

In this phase, all testers will provide two test IdPs.

For each SP in InCommon metadata, HTTP Client does the following:

1. If the SP does not support SAML2 HTTP-POST, skip to the next SP
2. Probe the SP for liveness by sending an empty POST request to the SP's SAML2 HTTP-POST endpoint
3. If the SP does not respond, skip to the next SP
4. Push a SHA-1 signed SAML2 Response from IdP #1 to the SP's SAML2 HTTP-POST endpoint
5. Push a SHA-2 signed SAML2 Response from IdP #2 to the SP's SAML2 HTTP-POST endpoint
6. Compare the two HTTP responses
7. Continue with the next SP