SP Testing for SAML2

Testing a SAML V2.0 SP

This page outlines a strategy for testing an SP that supports SAML V2.0 Web Browser SSO. In the InCommon Federation, SPs support SAML V2.0 for maximum flexibility and interoperability.

A typical SAML flow involves multiple exchanges, most of them terminating at the SP. Since it is up to the SP to initiate the next exchange in the flow, most errors tend to originate with the SP. Ironically, many of these errors manifest themselves at the IdP, often as a result of faulty SP metadata. Thus it is critical that SP metadata accurately reflect the capabilities of the SP.

Once the SP metadata and software have been upgraded to SAML V2.0, a natural tendency is test the complete, end-to-end flow. If this works, you may be done, but if more thorough testing is required, a targeted test sequence may be employed:

1. Test the SP's ability to consume a SAML V2.0 assertion response (bypassing the SAML V2.0 authentication request)
2. Test the SP's ability to issue a SAML V2.0 authentication request (bypassing IdP discovery)
3. Test the SP's ability to issue a SAML V2.0 Identity Provider Discovery Protocol request (bypassing nothing)

The following sections show how to perform such tests to track down bugs and ready the deployment for production.

Phase 1 Testing

To test your software's ability to consume SAML V2.0 assertion responses, methodically push an unsolicited response to each configured `<md:AssertionConsumerService>` endpoint. Since an unsolicited response is initiated at the IdP, an explicit authentication request is bypassed, which focuses testing on the SP's ability to consume the response. To perform this test, you need the following information:

- The unsolicited SSO endpoint location at the IdP (which is not advertised in IdP metadata)
- The `entityID` of the SP
- The `<md:AssertionConsumerService>` endpoint location(s) at the SP

First initiate an unsolicited response for the *default* `<md:AssertionConsumerService>` endpoint location at the SP:

```
https://idp.protectnetwork.org/protectnetwork-idp/profile/SAML2/Unsolicited/SSO?providerId=https%3A%2F%
2Fservice1.internet2.edu%2Fshibboleth
```

In the previous request, we use the unsolicited SSO endpoint location at the ProtectNetwork IdP and the `entityID` of the InCommon Operations SP (suitably encoded). Your mileage may vary of course.

Subsequently test a *specific* `<md:AssertionConsumerService>` endpoint location at the SP by appending a `shire` parameter to the above query string:

```
...&shire=https%3A%2F%2Fservice1.internet2.edu%2FShibboleth.sso%2FSAML2%2FPOST
```

where the value of the `shire` parameter is a suitably encoded endpoint location at the SP. Continue to test *every* `<md:AssertionConsumerService>` endpoint location in this way.

Note that the HTTP parameters used to trigger an unsolicited response (`providerId` and `shire`) are the same parameters used in a Shibboleth 1.x `AuthnRequest`, but since the endpoint at the IdP is a SAML V2.0 endpoint, a SAML V2.0 flow is initiated.

Phase 2 Testing

Next test your software's ability to issue SAML V2.0 authentication requests by initiating a SAML V2.0 discovery response at the SP itself. Similar to the way we initiated an unsolicited response from the IdP in the previous section, we can initiate a discovery response at the SP. Two additional pieces of information are needed:

- The `<idpdisc:DiscoveryResponse>` endpoint location(s) at the SP
- The `entityID` of the IdP

Initiate a discovery response for an `<idpdisc:DiscoveryResponse>` endpoint location at the SP as follows:

```
https://service1.internet2.edu/Shibboleth.sso/DS?entityID=urn%3Amace%3Aincommon%3Aidp.protectnetwork.org
```

In the previous request, we use the (one and only) discovery response endpoint location at the InCommon Operations SP and the `entityID` of the Protect Network IdP. Again, your mileage may vary.

Repeat the previous test for each configured discovery response endpoint. Each such endpoint is associated with a different subset of `<md:AssertionConsumerService>` endpoints, so be sure to exercise each of them in turn.

Phase 3 Testing

In this final round of testing, we trigger a response from a discovery service endpoint, which is typically the first point of contact in SP-initiated Web Browser SSO. Since the above tests have shown all exchanges but the first to work correctly, this final test has a high probability of success (or at least we know where to look if it doesn't).

We will use the InCommon Discovery Service for this test:

```
https://wayf.incommonfederation.org/DS/WAYF?entityID=https%3A%2F%2Fservice1.internet2.edu%2Fshibboleth
```

However, the above request causes an error due to a bug in SWITCHwayf 1.14. Until that bug is fixed, issue the following request instead:

```
https://wayf.incommonfederation.org/DS/WAYF?entityID=https%3A%2F%2Fservice1.internet2.edu%
2Fshibboleth&return=https%3A%2F%2Fservice1.internet2.edu%2FShibboleth.sso%2FDS%3F
```

In the above request, the value of the `return` parameter is a specific `<idpdisc:DiscoveryResponse>` endpoint at the SP (with a '?' tacked on to the end). Try each `<idpdisc:DiscoveryResponse>` endpoint in turn to complete this last series of tests.