UnsolicitedSSOConfiguration

File(s): *conf/relying-party.xml*

Format: Native Spring / Deprecated Custom Schema

Legacy V2 File(s): *conf/relying-party.xml*

Overview

Usually in SAML and most similar SSO protocols, the flow is assumed to be a service requesting authentication by redirecting the client to the login service, and then getting back a response. In the original SAML 1.0 and SAML 1.1 standards, though, SSO was described in only semi-interoperable terms as a "pushed" response from the IdP to the SP, and the "request" portion was left out. This was carried into SAML 2.0 as a mode called "IdP-initiated" or "unsolicited" SSO.

While this approach lacks interoperability, it has perceived benefits for some service providers; they get to do less work and push that work onto users and IdPs. So it isn't unusual to find SPs that refuse to support the standard fully and insist on this approach.

What is misunderstood about this feature is that it is **not** interoperable, despite being part of the standard. Interoperability requires a well-defined message, and the basic idea behind IdP-initiated SSO is that the message is up to the IdP. **Something** has to initiate the process, it can't magically start for no reason. So there is a request to the IdP, but it isn't a SAML-defined message and no two IdPs are likely to work the same way. It's a proprietary mechanism.

Configuration

There is no special configuration for this use case, it's subsumed into supporting Browser SSO for SAML 1 and SAML 2 for a relying party. In the default configuration, both are enabled using the profile configuration beans named "Shibboleth.SSO" and "SAML2.SSO".

There is currently no simple method of disabling the Unsolicited SSO support for SAML 2 separately from the overall support for SAML 2 SSO. If you need to disable this feature, it's possible in V3.3+ to remap existing profile locations to your own flow definitions; you can inquire on the support list if you want to do this.

Another way you can disable support for this feature for specific services is by modifying their SAML metadata to include `AuthnRequestSigned="true"` in the `<SPSSODescriptor>` element. Doing so causes the IdP to require requests from that SP to be signed, and since this protocol does not allow for signing, it will cause such requests to fail with an error.

Request Interface

As described above, the protocol for this feature is proprietary and is defined by the Shibboleth software. The interface to ask the IdP to respond to an SP without the SP having made a request involves the use of some defined query string parameters.

SAML 1.x

In older Shibboleth versions, the lack of a request message in SAML 1.x was supplemented with a simple request format defined in the Shibboleth Protocol Specification.

Out of the box, requests are handled at **https://*hostname*/idp/profile/Shibboleth/SSO** (replacing *hostname* with the location of your IdP) and the following query string parameters can be used:

- `providerId`
    - the name (i.e., the entityID) of the service provider
- `shire`
    - the URL of the SAML 1.1 response location at the SP (called the "Assertion Consumer Service")
- `target`
    - a target resource at the SP, or a state token generated by an SP to represent the resource
- `time` (optional)
    - a timestamp to help with stale request detection

Formally speaking, this is **not** IdP-initiated SSO; it's a proprietary request to the IdP that results in a response to the SP. If you refer back to the initial discussion above, you can see that that's actually the definition of IdP-initiated SSO.

SAML 2.0

The original protocol was adapted so that it can be used to trigger SAML 2.0 SSO in addition to legacy SAML 1.x responses. The two protocols are **not** supported at the same location in the IdP; that is, you can't send the request to one location and have the IdP figure out which protocol to use. It's simply an alternative request format that requires the identified SP support SAML 2.0.

Out of the box, this endpoint can be found at **https://*hostname*/idp/profile/SAML2/Unsolicited/SSO** (replacing *hostname* with the location of your IdP) and the following parameters can be used:

- `providerId`
  - the name (i.e., the entityID) of the service provider
- `shire` (optional)
  - URL of the SAML 2.0 response location at the SP (the "Assertion Consumer Service"), but can be omitted in favor of the IdP picking the default ACS location from the SP's metadata
- `target` (optional)
  - corresponds to `RelayState` in the SAML 2.0 protocol, but can generally be omitted
- `time` (optional)
  - a timestamp to help with stale request detection

As you can see, this is the same protocol as before, but with more optional parameters, reflecting how we would have designed the protocol if we were starting from scratch today. Protocol syntax is compatible with the original so that existing links can be easily adapted and used, despite the fact that the terminology is outdated (you don't want to know the origin of the name "shire", but it doesn't involve hobbits).

Examples

The examples assume the default locations supported out of the box, which should rarely need to be adjusted, and an IdP located at idp.example.org.

They also assume proper metadata is loaded into the IdP. There is nothing special about the metadata required to use this feature, it's the same metadata required for any use of the corresponding SP, and if you have to create that by hand because you're working with a deficient partner and/or outside the context of a federation, see other topics and examples to explore how to do that.

Given an SP named "https://sp.example.org/shibboleth", requesting SAML 2.0 SSO to the SP's default endpoint in metadata is just a link to:

```
https://idp.example.org/idp/profile/SAML2/Unsolicited/SSO?providerId=https%3A%2F%2Fsp.example.org%2Fshibboleth
```

A real world example of a non-Shibboleth SP that also requires a target parameter to identify a specific service to invoke (the names have been sanitized to protect the ignorant):

```
https://idp.example.org/idp/profile/SAML2/Unsolicited/SSO?providerId=http%3a%2f%2ffederation.morons.com%2fadfs%
2fservices%2ftrust&target=rpid%3dhttps%253A%252F%252Ffederationx.morons.com%252FClaimsAwareHelper%252F%26wctx%
3dTWN-EE-ER"
```

The example above is something you'll run into occasionally: the doubly-encoded value. When an SP is sufficiently broken, it may require query parameters that are themselves passed as values of the target parameter (as shown above, where the target value is a query parameter named "rpid" and with the value of another URL). The URL parameter to which the "rpid" name is assigned is URL-encoded, and then the entire name/value pair is URL-encoded for inclusion in the "target" parameter.

It's not common to have to provide the "shire" parameter in the SAML 2.0 case, but it may be needed if a single SP has development and production versions distinguished by SAML endpoint; the parameter would tell the IdP where to send the browser after logging in, and thus which environment to access.