IdP Persistent IDs

Configuration changes and database migration

SWITCHaai Team
aai@switch.ch

Persistent IDs in SAML – short recap

- a *persistent, revocable, non-reassignable, opaque, targeted, non-global* identifier for identifying the subject in a SAML assertion
  [https://wiki.shibboleth.net/confluence/display/CONCEPT/NameIdentifiers]
- introduced in 2005 with the SAML V2.0 specification:
  "Persistent name identifiers generated by identity providers MUST be constructed using pseudo-random values that have no discernible correspondence with the subject's actual identifier (for example, username). The intent is to create a non-public, pair-wise pseudonym to prevent the discovery of the subject's identity or activities."
- first implemented in the Shibboleth IdP 2; full-featured persistent ID support requires a database
- configuration instructions included in the SWITCH IdP deployment guide since 2008, with MySQL as the suggested backend

Persistent IDs in practice

- on the wire (in a SAML assertion)
```
<NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"
NameQualifier="https://aai-login.example.org/idp/shibboleth"
SPNameQualifier="https://sp.example.org/shibboleth">jQ+KQZR4OHyNi9702/
kW5KIQFhk=</NameID>
```
- attribute rendering in the Shibboleth SP (string)
```
https://aai-login.example.org/idp/shibboleth!https://sp.example.org/
shibboleth!jQ+KQZR4OHyNi9702/kW5KIQFhk=
```
- by default, the Shibboleth IdP creates the persistent ID proper by calculating the SHA-1 hash of the SP's entity ID plus a user attribute value plus an admin-specified salt (i.e., the output is 20 bytes, Base64 encoded)
- when used in a federation, always qualified by the IdP and SP entity IDs

Persistent ID changes with the IdP v3

- no disruptive ones, but a couple of things have happened behind the scenes
- most importantly, the IdP v3 brings a **new, dedicated NameID generation service**
  - preferred over the previous method available in v2, which treated name IDs as a sort of "special-purpose" attributes
  - deprecates the **`StoredId`** data connector and the **`SAML2NameID`** attribute definition type
  - in a pure v3 configuration and an ideal SAML 2 world, the IdP would only include NameIDs in the **`<Subject>`** element of an assertion
  - the configuration in the v3 SWITCH deployment guide has been updated to the new-style generation as far as possible, but still allows encoding of persistent IDs in SAML attributes

Configuring the NameID generation service

- configure the parameters for the service in **`/opt/shibboleth-idp/conf/saml-nameid.properties`**:

```
idp.persistentId.generator = shibboleth.StoredPersistentIdGenerator
idp.persistentId.store = PersistentIdStore
idp.persistentId.sourceAttribute = swissEduPersonUniqueID.withoutAttributeEncoder
```

- to enable the generating side of the service, remove the comments around the **`shibboleth.SAML2PersistentGenerator`** bean reference in /opt/shibboleth-idp/conf/saml-nameid.xml
- to support the reverse mapping (from a persistent ID back to a user), remove the comments around the **`c14n/SAML2Persistent`** bean reference in **`/opt/shibboleth-idp/conf/c14n/subject-c14n.xml`**
- finally, add the proper **`idp.persistentId.salt`** value to **`/opt/shibboleth-idp/conf/credentials.properties`** (carry over from your v2 configuration)

Retaining persistent IDs from your v2 IdP

- the database schema for the **`shibpid`** table remains unchanged
- when setting up a new IdP v3 from scratch, SWITCH recommends PostgreSQL as the database backend (unless relying on an existing, separately hosted RDBMS)
- “transferring” the records from MySQL to PostgreSQL is straightforward:

```
me@idpv2$ sudo mysqldump --compatible postgres --compact --no-create-info
                         --result-file shibpid.sql shibboleth shibpid
me@idpv3$ sudo -iu postgres psql shibboleth --file /path/to/shibpid.sql
```

- make sure to import the records into an empty table, i.e. execute **`sudo -u postgres psql shibboleth -c "truncate shibpid"`** before importing a newer version of a full dump

(Some ideas for) hands-on exercises

- examine the current contents of the **`shibpid`** table:

```
$ sudo -iu postgres psql shibboleth
shibboleth=# \pset pager off
shibboleth=# \dS shibpid
shibboleth=# select * from shibpid;
```

- delete a record from the **`shibpid`** table, and “recreate” it by logging in again with the respective account (on the proper SP)
- dump the **`shibpid`** table to a file, purge the table with truncate, and reimport the records
- log in to your v2 [test] IdP, figure out the current number of **`shibpid`** records, and make a dump of that table