Attributes für Apps
Eine App-gefahrene Lösung für AAI auf Smartphones

Lukas Hämmerle
lukas.haemmerle@switch.ch

AAI Forum, Berlin, März 2014
Einführung

- Universitäten bieten Apps, z.B. für e-Learning und Campus-Info
- Apps benötigen Authentifizierung
- Apps meist nicht-browser-basierte Anwendungen
- Authentifizierung und Autorisierung Infrastrukturen (AAI) basierend auf SAML2 sind schwer zu verwenden für non-browser-basierte Anwendungen

App by University of St. Gallen
Wish List for a Solution

• Login for users from **many organisations** via AAI
  – Excludes authentication with LDAP or HTTP Basic Auth

• Solution for **full-mesh** (Shibboleth-based) federation
  – Authentication for Apps in hub-and-spoke federations is easier
  – Moonshot, OpenID Connect are not deployed yet

• **No changes**/updates/plugins for Identity Provider needed
  – Excludes SAML Enhanced Client and Proxy (ECP) profile

**Solution that works today** (in SWITCHaai/DFN-AAI)!
App Requirements

• Not “emulate” a web-browser for authentication
  – Excludes already known approaches

• Not save password of AAI user
  – Would cause problems (app data stolen by other app, commercial company offering app, password change)

• Authenticate not too often
  – Apps should be easy to use and behave like other apps

• Always get up-to-date user attributes on start
  – Excludes approaches based on caching user attributes
Solution: SAML Attribute Query

- Service Provider (SP) directly queries Attribute Authority (AA) of Identity Provider for user attributes using a NameId.
- Usually transient or persistent NameID is used.
- Shibboleth SP/IdP supports Attribute Queries by default.
  - SWITCHaai federation: 98% IdPs and SPs are using Shibboleth.
Persistent ID and Attribute Queries

Serialized Example of a persistentId/ePTID:


• **PersistentID**: Persistent, Opaque, Targeted  
  Same user has different persistent IDs for different services.  
  Is unknown until user accessed service at least once.

• Once SP has user’s persistent ID, **up-to-date attributes**  
  can be queried **without user’s involvement** at any time!
  – Allows de-provisioning and account data updating
Architecture of Solution

- A (Mobile) Proxy translates authentication/attribute information from SAML2 to OAuth/REST/JSON
- Mobile Proxy includes an OAuth2 Server that grants access tokens, which are mapped to a SAML2 persistent ID
Concept of Mobile Proxy

1. User authenticates once at Mobile Proxy via web browser
2. Mobile Proxy gets persistent ID of user
3. Proxy stores persistent ID and binds it to an OAuth2 access token, which is stored in the App
4. App queries Mobile Proxy for AAI attributes with token
5. Mobile Proxy uses persistentId to query user’s AAI attributes via a SAML Attribute Query
User’s Perspective: First App Start

- User starts app for the first time
- App asks user to authenticate with AAI on device or desktop PC
- Mobile browser opens and user selects his organisation
User’s Perspective: First App Start Continued

- Authentication with AAI at home organisation in web browser
- Mobile Proxy SP gets user’s attributes including persistentId and issues OAuth token
- Uni App uses token to get user attributes from Mobile Proxy

Link with custom URL scheme is opened automatically
E.g. uniapp://{App-Identifier}/{40-Byte-Access Token}
User’s Perspective: Further App Starts

- User starts app
- App fetches user attributes with OAuth access token from proxy
- App gets other app-specific data with access token
Mobile Browser vs Desktop Browser

To initially get persistent ID, user must login with a web browser at least once with AAI. Options:

• **In-App browser:**
  – In app browser might not have access to browser saved passwords user has to type in again username password at IdP

• **Browser on mobile device:**
  – Benefit from SSO session that user might have already
  – Default browser on device is used

• **Browser on Desktop:**
  – Most flexible browser that might support authentication methods other than username/password. E.g. X.509
  – Requires user to type URL/token or scan a QR code
Data Flow

Initial Web SSO authentication with browser

1. persistentId: sdf9823nou

Attribute Query to:
/idp/profile/SAML2/SOAP/AttributeQuery + sdf9823nou (persistentId)

2. Mapping Table
   - Token: 2$892
   - persistent ID: 23cj3r0hw
   - 69m.i: asd823enc
   - 4k@8: sdf9823nou

3. SP

4. AA

5. JSON/XML User record

6. Uni App

Querying personal data and settings...

App-specific service

Request to get additional data from resource service with OAuth access token

App-specific user data
App Logout / Access Token Revocation

How about revocation of OAuth access token? For example in case the device is sold or lost.

• OAuth Access token is used to:
  – Authenticate with Mobile Proxy
  – Retrieve up-to-date AAI attributes from Mobile Proxy
  – Retrieve arbitrary protected resources from third party resource server

• Token can be revoked by:
  – Expiration because validity is configurable
  – User within App by clicking on “Logout”
  – User via administration interface with web browser
  – Admin of Mobile Proxy
Logout/Token Revocation via Web Interface

Mobile Proxy Device Administration

Multiple devices for same user and same app

Authenticated user
Do-it-yourself Demo of Sample Uni App

• A quick demo is available on the AAI for Apps web page: https://www.switch.ch/aai/support/tools/aai-for-apps.html

• Two options for initial AAI login:
  – Browser on mobile device
  – Browser on another computer (requires typing or scanning QR code)
Disadvantages and Technical Requirements

• Not all IdPs support persistent ID with stored ID
  – Identity Provider must have an Attribute Authority
    (is the case for 98% of all IdPs in DFN-AAI, state 4.3.2014)
  – Attribute queries don’t work with computed (hashed) persistentIDs
    (unknown if this is the case for the IdPs in DFN-AAI)

• Shibboleth SP protecting Mobile Proxy:
  – Must be Shibboleth 2.5 or newer
  – Requires Attribute Query Extension by NII/GakuNin that allows fast
    Attribute Queries via Shibboleth handler.

• Sample App framework currently available only for Android
  – iPhone version might follow
Advantages of this Approach

• App never gets user’s AAI username/password
  – Any type of authentication can be used for initial web authentication!

• Can be deployed immediately without changes to federation
  – Requires that IdPs support persistentId (with storedId) and attribute queries. This is the case for all SWITCHaai IdPs.
  – Approach also works when SP aggregates attributes from additional attribute authorities (Virtual Organization/Group attribute providers)

• One instance of Mobile Proxy can serve multiple apps
  – Apps can have different attribute requirements
  – Individual &lt;EntityDescriptors&gt; for each app possible
Availability and Future Plans

- Software available as Open Source software (BSD license)
  - **Sample Uni App**: Java, Android App ready for customization
  - **Mobile Proxy**: PHP, Includes OAuth server and simple web interface
  - **Resource Server**: PHP, Returns back a default time table

- Developed as Prototype. No production quality yet.

- More information and link to SVN repository: https://www.switch.ch/aai/support/tools/aai-for-apps.html
Credits and References

• Daniel Latzer
  SWITCH apprentice who did most of the implementation

• NII/GakuNin (JP)
  Created the Attribute Query extension for Shibboleth
  (not public yet)

• Similar Ideas/Approaches:
  – “FACIUS: An Easy-to-Deploy SAML-based Approach to Federate Non Web-Based Services”, Steinbuch Centre for Comput. (SCC), Karlsruhe Inst. of Technol. (KIT), Karlsruhe, Germany
  – University of Malaga