24-06-2016

Milestone MJRA1.3: Design for the integration of an Attribute Management Tool

Milestone MJRA1.3

Contractual Date:	30-04-2016
Actual Date:	24-06-2016
Grant Agreement No.:	653965
Work Package:	JRA1
Task Item:	JRA1.4
Lead Partner:	GARR
Document Code:	MJRA1.3
Authors:	D Vaghetti (ed), M-L Mantovani, Andrea Biancini (GARR), N Liampotis, C Kanellopoulos (GRNET), J Jensen (STFC), U Stevanovic (FZJ), P Birkinshaw, M Haase (DAASI), P Solagna (EGI), M Jankowski (PSNC), M Procházka, S Licehammer (CESNET)

The research leading to these results has received funding from the European Community's Horizon2020 Programme under Grant Agreement No. 653965 (AARC).

Abstract

In the present work we analysed Attribute Authorities delegation and aggregation capabilities, and Attribute Translation Mappings among different protocols in order to implement an integrated and interoperable authentication and authorisation infrastructure (AAI) for the Research and Educational (R&E) community. The two main issues addressed are the limitations in defining attributes and VOs for research communities in their home organizations, and the added complexity of a multi-protocol digital identity landscape.

Table of Contents

Table of Figures

Table of Tables

Executive Summary

In this Milestone, tools and technologies that can be used to implement an integrated and interoperable Authentication and Authorisation Infrastructure (AAI) for the Research and Educational (R&E) community have been analysed.

SAML-based National R&E Identity Federations and the eduGAIN interfederation provide a solid foundation for the building of interoperable AAIs. However, the digital identity landscape is more complex: on the one hand there are important research communities that make use of different authentication protocols and standards, the most prominent being X.509; on the other hand there are pervasive social identities, mainly based on OAuth2 and OpenID Connect, and the eGOV-ID initiatives based on different implementation of SAML. The design of an integrated AAI for the R&E community must take into account such a scattered and multi-protocol scenario, and to achieve that goal attribute translation and mapping among different protocols has been analysed.

Another important aspect is the authorisation layer supporting access to resources, which is typically attribute- or role-based. Given the inter-organizational nature of the R&E community, attributes and roles (along with user identities) often come from different home organizations. Moreover, research communities that have the need to define custom attributes cannot always leverage their home organization Identity Management Systems because of policies or technical issues. To overcome such limitations, we analysed Attribute Authorities by their delegation and attributes aggregation capabilities:

- AAs that can enrich the attributes set available to the authentication flow.
- AAs with a management interface that permits autonomous management of attributes and/or delegation to subjects external to the home organization.
- AAs with attributes aggregation and manipulation capabilities.

It should be noted that the AAs analysis has not been limited to the SAML landscape, but also covered the X.509 standard, paying particular attention to multi-protocol support. At the same time implemented attribute flow models have been compared, and prominent issues pointed out. No recommendation for the adoption of a particular tool has been made, nor is there a simple table of feature comparisons (these can be found in the work of the pilot tasks of the AARC project), but details about the implementations of the reported features have been presented, together with, where possible, actual use cases.

The work on attribute translations and mapping has taken into account the following use cases:

- SAML to OpenID Connect (with coverage for social identities).
- SAML to X.509.
- SAML-to-SAML (eGOV-ID).

In addition, mapping of group membership information to authorisation has been analysed.

Attribute releasing is a key factor in an integrated and interoperable AAI, so recommendations have been made for ways to improve it, leveraging Research and Scholarship Entity Category and providing the user with more information in the case of a failure to access resources.

This Milestone includes an analysis and comparison of many different aspects of an integrated and interoperable AAI. Important topics left out, like Token Translation Services, Non Web-Access, and Delegation of authorisation, will be addressed in AARC MJRA1.4 'First Draft of the blueprint architecture'.

1 Introduction

This Milestone has analysed Attribute Authorities' delegation and aggregation capabilities, and Attribute Translation Mappings among different protocols, in order to implement an integrated and interoperable authentication and authorisation infrastructure (AAI) for the Research and Educational (R&E) community. The two main issues addressed are the limitations in defining attributes and Virtual Organizations (VOs) for research communities in their home organizations, and the added complexity of a multi-protocol digital identity landscape.

Among the works on related subjects, the following have been used as references for the present Milestone:

- AARC 'DJRA1.1 Analysis of user-community requirements'[1]
- AARC 'MJRA1.1 Existing AAI and available technologies for federated access'[2]
- GÉANT GN4-1 JRA3 T1 (Attributes and Authorisations). Deliverable 'D15.1 Report on the Achievements for T1 and Recommendations on Future Work'[3])

[1] https://aarc-project.eu/wp-content/uploads/2015/10/AARC-DJRA1.1.pdf
[2] https://aarc-project.eu/wp-content/uploads/2016/01/MJRA1.1-Existing-AAI-and-available-technologies.pdf
[3] http://www.geant.org/Projects/GEANT_Project_GN4-1/Documents/D15-1_Report-on-the-Achievements-of-JRA3-T1-and-Recommendations-on-Future-Work.pdf

2 Attribute Authorities

2.1 Models

2.1.1 Push vs Pull

In the following discussion of attribute release and aggregation, it may be helpful to think of "attributes" as a set of name/value pairs contained in an "envelope" which may be digitally signed by the Attribute Authority that issued it, and to think of the identity provider (IdP) as an AA as well. So an IdP might create a set of attributes `{email=joe.user@example.com; organisation="Example Ltd"; role=staff}` and a community AA may create a set `{user=joe.user@example.com; status=member; joined=20140203172314; role=admin}`.

While it is currently rare that users need attributes from more than a few (one or two) AAs, the need for combining attributes from more AAs may become more frequent. In fact, such a situation is desirable when cross-community collaborations are to be fostered and roles need to be combined from different communities, or when users access multiple infrastructures at the same time (with the same credential) and need to present several sets of attributes. In these cases, the push/pull models differ in how they respond to the number of AAs.

For the push model, users will normally obtain the attributes they need and push them to the Service Provider (SP) every time they use their credential. The attributes will typically be cached at the SP locally and will have a validity lifetime mandated by the AA, of typically 12 hours. In the pull model, each SP must obtain the credentials upon successful user authentication and must cache them itself.

PUSH

- SAML (WebSSO): The user is directed to an IdP where attributes are obtained; i.e. the IdP acts as an AA in the sense that it can *augment* the assertion with extra attributes, which the IdP maintains or has found elsewhere. The user/client pushes this attribute assertion back to the SP. In case the user is directed to an SP-IdP-Proxy, the same workflow happens. The SP-IdP-proxy typically provides a set of Attributes, which was previously aggregated (e.g. by Attribute-Aggregation, Group Admin or by self-assertion).
- OpenID Connect: The OIDC has the option to return attributes from the authorisation server to the client after successful user authentication. While this will normally send an authorisation code (for Authorization Flow) or an ID token/Access token for Implicit Flow, it can also send other attributes, and should normally ask for user consent after authentication and before returning attributes to the client. The consent decision may be cached; the client can then decide to forward this information to the resource server.
- VOMS (Virtual Organization Membership Service): The user contacts the AA (namely the VOMS server) with its credential and obtains a delegated credential with authorisation attributes. The delegated credential (see also Section 5 on delegation) is generated locally, by the user. This contrasts with the actual attribute assertion (usually an RFC3281 attribute certificate, but SAML could be used instead) which is signed by the VOMS server and included in the certificate as an extension. There can be several

servers, but the user provides hints for discovery. Only trusted VOMS servers are acceptable; their metadata is deployed throughout the infrastructure, so an attribute from an untrusted VOMS server would not be honoured.

- X.509: The user is issued with a signed assertion (i.e. a certificate proxy) which is sent to the server upon every connection. VOMS is a special case of this scenario, since the user generates a proxy certificate with the VOMS assertion inside it. Note that basic X.509 only has the Certificate Authority (CA) as the issuing authority, there is no additional AA.

PULL

- SAML 2.0 (Attribute Query): The user requires access to a service and initiates a federated single sign-on flow. The IdP authenticates the user and issues a SAML assertion with a set of attributes. The SP then issues an AttributeQuery to (one or more) preconfigured Attribute Authorities in order to obtain additional attributes that can be aggregated with the initial set returned by the IdP. SAML attribute queries are particularly useful when there is an existing SAML identity federation with established policies and trust. Moreover, the additional user attributes are retrieved during the single sign-on process and merged with the IdP-provided attributes. Thus the end services can receive them in a standard SAML format without the need for any translation mechanism.
- OpenID Connect: OIDC defines the UserInfo Endpoint (an OAuth 2.0 protected resource that can be used to return information or Claims about users after they have been authenticated). To obtain the requested Claims about the user, the client makes a request to the UserInfo Endpoint using an access token obtained through OpenID Connect Authentication. These Claims are normally represented by a JSON object that contains a collection of name and value pairs for the Claims. Apart from the set of standard[4] Claims defined in the OIDC Core specification 1.0, additional Claims not specified there could also be returned.
- XACML: An XACML context handler or Policy Enforcement Point (PEP) queries a PIP for attributes using a SAML attribute query. The requester asks for attributes by name (as opposed to the PIP sending everything). There exists a SAML profile which "defines how to use SAML 2.0 to protect, store, transport, request, and respond with XACML schema instances and other information needed by an XACML implementation"[5].

2.1.2 Mesh vs Proxy

There are different models of aggregating attribute flows among SPs and AAs: these may be distinguished as Mesh and Proxy:

Mesh: Each and every Service Provider contacts relevant AAs and collects attributes for the user. For this to work, a certain amount of negotiation needs to take place to establish technical connections and trust. This model works well where there is little trust between entities, and it has excellent ways of preserving privacy. The cost of this is in added complexity as multiple technical and trust relations need to be maintained between multiple parties.

Proxy: A single entity, a proxy, collects attributes at relevant sources on behalf of the Services, and then passes the combined set towards the Service. For this to work, a central entity must be operated and both services, as well as IdPs, need to trust the operator of the entity. In VO scenarios, it would be feasible for the VO to operate

[4] http://openid.net/specs/openid-connect-core-1_0.html#StandardClaims
[5] http://docs.oasis-open.org/xacml/3.0/xacml-profile-saml2.0-v2-spec-en.html

the proxy. The proxy takes away a lot of the complexity, as the number of interactions on the technical and trust level are reduced to connecting to the central proxy for all entities.

2.2 Identity Providers as Attribute Authorities

2.2.1 Tools and Technologies

2.2.1.1 *Shibboleth*

The Shibboleth IdP v3.X attribute releasing functionalities are provided by a set of components collectively responsible for attributes retrieval, processing, filtering, and encoding. At the end of the process the attributes, encoded in a protocol specific representation (usually SAML2 statements), are sent to the Relying Party.

The two main components are the Attribute Resolver and the Attribute Filter Engine. The Attribute Resolver is in turn made of three subcomponents:

- Data Connectors: the components responsible for raw attribute collection from source systems like LDAP and RDBMS (JDBC). Other data connectors have been developed by the community and are available in the form of extensions to the Shibboleth IdP, though many of them are not yet available for the latest release of the software.
- Attribute Definitions: the components that associate encoders, display names and descriptions to an attribute. Some kind of transformational logic is usually performed on the attribute's values by a range of native functions (the attribute definitions themselves), or by custom transformation functions written in JavaScript (`ScriptedAttributeDefinition`).
- Attribute Encoders: the components that encode the transformed attributes to their protocol specific counterparts.

The Attribute Filter Engine applies attribute filter policies to determine which attributes are to be released or not released.

The Shibboleth IdP attribute authority usually works with the Authentication Engine of the IdP to provide an attribute statement, along with the authentication statement to be sent to the requester. However, the AA can be engaged independently via a back-channel Attribute Query directed to the SOAP endpoint exposed by the Shibboleth IdP. The SOAP endpoint for Attribute Query implements the SAML 2.0 Assertion Query/Request Profile.

2.2.1.2 *SimpleSAMLphp*

SimpleSAMLphp attribute releasing functionalities are implemented via Authentication processing filters. Authentication processing filters post-process authentication information received from authentication sources, where the latter not only provides authentication (actually you can have “authentication sources” that do not

perform any authentication, but are used as data sources), but also attributes to be mangled, filtered, encoded and finally sent to Relying Parties. The Authentication processing filters are also used to implement consent.

There are tens of predefined filters included in the SimpleSAMLphp distribution that can accomplish many tasks related to attributes mangling, such as:

- Adding new attributes (with merging/replacing capabilities).
- Attributes remapping.
- Simple attributes transformation (i.e. regular expressions, scoping).
- Attributes transformation via custom PHP code.
- NameID generations strategies.

New filters can also be added by the IdP administrators as a sub-class of the `SimpleSAML_Auth_ProcessingFilter` class.

Note that SimpleSAMLphp's low-level SAML library[6] supports the SAML 2.0 Assertion Query/Request Profile using synchronous SOAP binding. However, a default SimpleSAMLphp-based installation configured as an IdP does not expose the attribute query/request service endpoint out-of-the-box. There is a core SimpleSAMLphp module, namely `exampleattributeserver`[7], which needs to be tweaked to support this, since the provided functionality is rather limited. An alternative is to use the third-party Attribute Authority (AA[8]) module that supports more advanced features, such as customising the attribute provider metadata, as well as authenticating the received attribute queries based on SSL client certificates, XML message signatures, etc.

2.2.1.3 *ADFS*

Web Services Federation (WS-Federation) was designed to "facilitate brokering of trust and security token exchange," including the option to map identities between different federation participants at the Security Token Service (STS). Note that STS is used here in its WS-Federation sense; not in the sense of the specific implementation described in section 3.2.3. Thus, the user's identity might be encoded by their original principal (a "fixed identity" as encoded by the security token issued by the original IdP) or it might be mapped by the STS in order to harmonise the attributes carried onward to the SP – by adding attributes understood by all federation SPs – or to anonymise the user by removing the original security tokens and adding a pseudonym encoded in a security token issued by the STS itself. Depending on federation demands and policies, these pseudonyms, in turn, can be both targeted (at specific SPs) or common to a set of SPs; they can be issued per session or persistent across sessions.

Thus, the STS can ensure the harmonisation of security tokens as seen by the SPs within a federation of diverse IdPs. Indeed, the IdPs (the issuer of original security tokens) may be based on completely different underlying security protocols (e.g. Kerberos and X.509), but both are structured as security tokens as specified by WS-Security. The STS can implement not only federation policies of user identities (pseudonyms and persistence), but also roles and other additional attributes which may be used by SPs for accounting and access control purposes. Also, further STS services can be connected to enable security domains to inter-federate.

[6] https://github.com/simplesamlphp/saml2
[7] https://github.com/simplesamlphp/simplesamlphp/blob/master/modules/exampleattributeserver/www/attributeserver.php
[8] https://github.com/NIIF/simplesamlphp-module-aa

2.2.1.4 *UNITY*

General description

Unity claims to be a complete solution for identity, federation and inter-federation management, or in other words it claims to be a flexible authentication service[9]. Authentication can be performed using an internal database (then the service plays the role of identity provider), or it can be delegated to an external identity provider (then the service plays the role of proxy IdP or Token Translation Service).

Unity manages and authenticates entities that could be users or software agents. Each entity can have one or more identities of different types (for example username, X.509 DN or persistent ID), and local credential. Entities are connected to groups that may form hierarchies. Entities may have assigned a number of attributes.

The entity interfaces with Unity via a configurable set of endpoints (including Web Admin UI or SAML IdP) and low-level protocol bindings. Each endpoint has associated `authenticators:components` that retrieve and verify single credential such as a password or certificate. The verification may be done locally by checking against credential stored in the local Unity database, or remotely by checking an external service (e.g. LDAP, SAML IdP). An endpoint may be configured to require multi-factor authentication.

Unity as IdP

The simplest way to deploy Unity as an IdP is by using local authentication. The user credentials are stored locally and the authentication is straightforward, as it always results in one of two states: valid or invalid. The setup may be supplemented by registration forms that may be used to allow users to apply for the account and collect some information from them. The registration may be configured to require remote authentication of the user. The registration may be accepted automatically or manually by the administrator. Unity can be configured to allow locally stored credentials to be reset if passwords are forgotten.

Another option to deploy Unity as an IdP is by using remote authentication with an LDAP server. This could be useful if a LDAP database is already in place (e.g. used for local authentication) and cannot be replaced by the internal Unity database (even though Unity may expose an LDAP interface). The user shall be automatically registered in Unity during the first login.

Currently, the following endpoints can be configured: web SAML2 SSO, web service SAML2 SOAP, SAML2 PAOS ECP, OAuth2 and OpenID Connect.

2.2.1.5 *Azure AD*

Azure AD (Active Directory) is Microsoft's current IdMaaS solution for linking organisational IdMs with the cloud (where the cloud is Microsoft Azure). It provides SSO in the sense that users log in with their home ID, but it also enables the home organisation to provide attributes to the cloud service for authorisation purposes (provided the home organisation runs AD internally), or if permitted, to allow users to look up other users. Optionally the cloud IdMaaS can link back to the organisational AD IdP and provide services for it such as password reset or security monitoring, or attributes in AD itself, so it extends AP from the organisation into the cloud. However, for the external IdP, it will also handle attributes from SAML and WS-Federation IdPs.

[9] http://www.unity-idm.eu/documentation/unity-1.8.0/manual.html

As far as the cloud app is concerned, it sees an OIDC sign in. Attributes are optionally available via the "Graph" API, which is REST-based and OData-compliant, with JSON responses that can optionally return XML.

From the perspective of this report, the main feature is that Azure AD enables the use of organisational attributes in a manner which is consistent with other attributes, such as those queried from the user.

2.3 Attribute management delegation

2.3.1 Tools and Technologies

2.3.1.1 *VOMS*

The Virtual Organization Membership Service (VOMS) is an Attribute Authority that asserts community specific attributes for users, in the form of extensions[10] of X.509 proxy certificates.

It is actively developed within INFN. VOMS is used in EGI for authorisation purposes, both to access HTC and Cloud services, serving as a repository for VO authorisation information and providing support for managing the virtual organisation membership and internal hierarchy.

VOMS allows communities to manage their structure and membership independently. VOMS provides administrators with the ability to organise users in groups, so called virtual organisations (VO), and designate specific roles and custom attributes to users. These capabilities are available both through an API and a GUI.

A single VOMS service can support many VOs, the communities may deploy their own instances, but it is more common among VOs that VOMS is operated by a resource provider on behalf of the community. Using a VOMS instance for multiple VOs also makes the use of resources more efficient, especially with high availability configurations.

Currently VOMS can provide the following information about a user:

- VO Membership.
- Group membership within the VO.
- Role ownership associated with the VO.

This user information is added as extensions to a user proxy certificate, and signed by the VOMS. In order to verify the proxy signature, SPs can make of use of LSC ("list of certificates") files whose contents only need to be updated when the certificate subject or the CA subject for a VOMS server has changed. In the past, a local copy of the host certificate for each trusted VOMS server had to be stored locally.

[10] RFC3820 Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile. S. Tuecke, V. Welch, D. Engert, L. Pearlman, M. Thompson. June 2004. (Format: TXT=86374 bytes) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC3820)

Usually VOMS extensions have a short validity (typically of 12 hours), and the user is authorized based on the information included in their proxy certificate extensions, even if the user's membership status has changed in the VOMS.

2.3.1.2 *Grouper*

Grouper is an enterprise access management system. It enables the creation and management of groups, roles, permissions and custom attributes that can be used to make authorization decisions.

Key features of Grouper are:

- Allow non-technical users (like researchers and administrative staff) manage access rules.
- It is designed for delegation and distributed management.
- Groups creation and membership management can leverage multiple sources, both automated or manual.
- All kinds of access policies can be expressed by the use of simple boolean logic, or more complex and *ad hoc* rules.
- It can be integrated with almost any existing IAM solution.

Grouper sports a great number of connectors for data sources (e.g. LDAP and SQL) and interfaces to expose group information and membership. Moreover, Grouper Web Services exposes common business logic through SOAP and REST calls.

Apart from the Grouper Web Services, another very common way to expose group information is by leveraging the LDAP Provisioning Connector that provisions group and membership information to an LDAP Directory. A common way, and by far the simpler, is to expose groups through a Shibboleth IdP by provisioning group information to the LDAP Directory configured as a data source for the Shibboleth IdP and configuring an attribute resolver for the isMemberOf attribute.

2.3.1.3 *COmanage*

The COmanage (Collaborative Organisation Management) platform provides a person registry designed to meet the needs of VOs with identity and access management requirements. Specifically, the platform includes:

- The COmanage Registry, an Identity Management System designed for Collaborative Organisations (COs).
- A set of REST APIs to connect federated applications that consume attributes and access control information provided by the COmanage Registry.

Some of the key features provided by the latest stable release (v1.0.3) include:

- Configurable user enrolment flows that support basic self-service registration, as well as more advanced enrolment processes (for example requiring email verification, approval by administrators, and so on.).
- Federated Identity: Users get authenticated at their home institution's IdP. Any attribute configured to be provided via SAML becomes organisational-authoritative and cannot be changed by the enrolee.

- Account linking to connect multiple institutional and/or social identities for the same person, either through self-service or by an administrator.
- Automatic identifier assignment, based on configurable rules.
- Automatic expiration of accounts.
- Configurable notifications around enrolment, expiration, and status changes.
- Delegated management of users via the Collaborative Organisational Units' (COU) hierarchical model, which is similar to LDAP OUs. Administrators can add roles and manage attributes for participants within their COU.
- Group management: Basic group management capabilities are directly provided by COmanage while more complex ones can be supported via the Grouper integration plugin.
- Various plugins and provisioners to applications that extend the functionality of COmanage, including Grouper, LDAP, and GitHub.
- Experimental VOOT support.
- SSH key management capabilities.
- Collaboration/VO-specific attributes to extend user information with attributes that may not be available from their home organization.
- Terms and Conditions functionality for VOs that have legal requirements around participation.
- Database support: PostgreSQL, MySQL / MariaDB, SQL Server (supported but not tested).
- Extended functionality/customisation through the use of plugins.

The COmanage project is funded by the US NSF[11] and Internet2[12]. The platform is being used in production at several sites and can be integrated with other collaboration platforms such as Grouper and/or OpenConext. The software is based on the CakePHP development framework. Installation guides, best practice information, and architecture information are available on the COmanage wiki[13]. The source code is available on GitHub[14], licensed under Apache2.

2.3.1.4 *Perun*

Perun[15] is an Identity and Access Management system. It is a complex tool which covers management of the whole ecosystem around user identities, groups and access management. It is designed to be flexible and customizable so it can be easily integrated with other tools or incorporated into existing workflows. Perun can obtain users' identities and groups by importing them from another system (such as. LDAP, SQL database, and XML files) or users can self-register using external identities (for example federated identity, X.509 certificate or social identity.). Users' identities can be linked/consolidated, so users can utilize any of their identities to access services managed by Perun. Having such mechanisms, Perun is able to manage user attributes and even aggregate attributes from different sources. A user's entry in Perun can be enriched by additional attributes.

Users are organized into groups and virtual organizations (VOs). Groups are used to control access to services. Perun can automatically prepare an access control list (ACL) file for the service each time any related data

[11] http://www.nsf.gov
[12] http://www.internet2.edu/vision-initiatives/initiatives/trust-identity-education-research/middleware-initiative-information
[13] https://spaces.internet2.edu/display/COmanage/Home
[14] https://github.com/Internet2/comanage-registry
[15] https://perun.cesnet.cz

(configuration, membership, etc.) is changed. Single point of failure is eliminated by propagating these ACLs directly to services (a PUSH mechanism). There are also other interfaces that can be used to query (a PULL mechanism) data about users, for example: LDAP, SAML2 Attribute Authority, VOOT and SCIM.

Perun software is maintained by CESNET and Masaryk University, Czech Republic, and it is open source software available under BSD-2 licence. Perun is used in Czech e-Infrastructure, in ELIXIR and in EGI.

2.3.1.5 *REMS*

REMS (Resource Entitlement Management System) is a tool to manage access rights to resources, in particular scientific datasets. REMS is effectively a workflow engine to which an Applicant logs in (using their SAML2 IdP) and identifies the resources (datasets) for which they want to apply for access rights. REMS asks them to fill in an application form configured for the resource, attach any relevant attachments (such as a research plan) and commit to any licence terms (such as a Data Access Agreement or a Material Transfer Agreement) the resource requires.

REMS then circulates the application for approval to the individuals (such as the dataset owner) or bodies (such as a Data Access Committee) configured for the resource. There can be several parallel or consecutive approvers defined for the resource, and the approvers can approve or reject the application, ask the Applicant to refine the application or forward the application to a third party for an expert opinion. All these end users use their SAML2 Identity Provider for log in.

Once an application is approved, REMS delivers the list of users and the resources permitted for them (effectively, an eduPersonPrincipalName attribute and the list of resource identifiers such as PIDs) to an external Policy Enforcement Point for enforcing the access rights. REMS provides also necessary reporting and audit trail functionality for access rights granted.

From the SAML2 perspective, REMS software is:

- A SAML2 Service Provider, to which both Applicants and Approvers can log in using their SAML2 Identity Provider.
- A SAML2 Attribute Provider, which responds to SAML Attribute Queries by providing a list of resource identifiers (for instance, a multi-valued eduPersonEntitlement attribute) permitted for the requested user (identified by eduPersonPrincipalName). REMS has a plug-in API for developing also integrations other than SAML2 (for instance, for pushing entitlements on SCIM).

The REMS tool is developed at CSC - the Finnish IT Center for Science as part of the ELIXIR research infrastructure projects and licensed under the LGPL license. The REMS tool is used, among other things, to manage access rights to sensitive human datasets in the ELIXIR research infrastructure, to biobank samples in the BBMRI research infrastructure and language datasets in the CLARIN research infrastructures. For more information, see https://www.csc.fi/rems.

2.3.2 Use cases

2.3.2.1 *DARIAH AAI*

The DARIAH AAI is a Shibboleth Idp as described in section 2.2.1.1 but with the following key differences:

- It uses a PULL model by leveraging the Shibboleth SP's SimpleAggregation feature.
- AA discovery is realized by static configuration.
- New users are redirected to a registration page (a self-service portal which receives attributes from the home organization via a frontend SP). This adds an entry in the AA's directory server, so that the AA releases all needed attributes upon a second query.
- Different attributes are provided by IdP and AA: all attributes and values are only accepted from the AA directly. Users are requested to keep their profile current, using the self-service; there is no need to update attributes during task execution.
- The scaling problem of an AA is not an issue yet – the IdP will be able to handle requests well in the foreseeable future.
- Trust of the SP and the AA is static via metadata.
- The SP is not restricted to get attributes for a different user.

The DARIAH IdP/AA will release further attributes to the service via direct SAML Attribute Queries, among them are:

- Personal data is needed but not released by campus IdP (e.g. researchers' mail).
- Approved terms of use for DARIAH, and service-specific ones.
- Authorization attributes (group memberships in central directory).

The last point in particular allows the AA to influence access decisions on the service, by delegating this right to administrators of the central directory.

2.4 Attribute aggregation service

2.4.1 Tools and Technologies

2.4.1.1 *Shibboleth SP*

Shibboleth provides the *Query* plugin to support aggregation of attributes from multiple attribute authorities through the use of SAML 2.0 attribute queries. These queries are based on an identifier derived from the attributes obtained after initial authentication against the user's IdP. Specifically, this simple form of aggregation relies on a plain-text identifier that can be used as a *link* to the user's account at other SAML attribute authorities. Conceptually, this mechanism works similarly to LDAP or X.500 directory queries by DN. No user intervention is required, which is an advantage in terms of simplicity but may be an issue for privacy and user control.

After each query is performed, Shibboleth applies the attribute extractor and filter configured for the application before continuing with other queries and eventually returning the resulting attributes. Each filtering step operates on only the attributes extracted as a result of a particular query.

2.4.1.2 *SimpleSAMLphp*

SimpleSAMLphp relies on third-party modules to provide attribute aggregation functionality: such modules are typically implemented as authentication processing filters. These filters are intended to post-process the user information received from the configured authentication sources. As such, they can be used to modify or enrich the attributes received about the user before delivering them to the SP application. Examples of SimpleSAMLphp modules that make use of the authentication processing filtering API to support attribute aggregation include:

- VOOT groups module[16]: Fetches group memberships from an API service protected with OAuth 2.0 using the VOOT protocol (versions 1 and 2 are supported) and adds them to the list of attributes received from the IdP,
- Attribute Aggregator module[17]: Issues SAML 2.0 Attribute Queries to Attribute Authorities that support the SAML 2.0 SOAP binding.
- Attribute-from-rest-api module[18]: Uses PHP's libcurl library to connect to a REST API that can provide information about the authenticated user. The retrieved user information is expected to be formatted in JSON for it to be decoded into attributes that can be added to the original list of attributes received by the IdP.

It should be noted that SimpleSAMLphp can be configured to execute more than one authentication processing filter (in series) and is thus able to aggregate attributes from multiple attribute providers.

2.4.1.3 *Unity/B2ACCESS*

Unity (see section 2.2.2.4) can be used not just as an AA in its own right but also to aggregate attributes from IdPs. Indeed, B2ACCESS (which consists of Unity and a CA, plus some additional components) is capable of picking up attributes from IdPs – such as email addresses – or to ask users to supply them if the IdP does not provide them. A harmonisation approach that sends only those attributes on to the SP that are provided by every single IdP is clearly not sufficient since there is in general not a single attribute that is sent by every IdP. Thus, B2ACCESS must aggregate the "missing" attributes.

It is possible for a B2ACCESS service to aggregate attributes for a single user from more than one IdP. This can be done only by the user themselves actively linking the IdPs to their existing account. Thus, a scheme could be imagined where a user registers with a high-LoA IdP and accepts the AUP, yet links in their lower-LoA social media or community (say) ID for subsequent ease-of-use (as the login is cached by the browser). The LoA used subsequently will in general be the most recently used, so in this example the lower one.

Following up from this, B2ACCESS then has requirements for (a) managing the LoA of individual attributes (for example an email address - was it provided by a high-LoA IdP, by a low-LoA one, by the user, was it verified?),

[16] https://github.com/OpenConextApps/ssp-voot-groups
[17] https://github.com/NIIF/simplesamlphp-module-attributeaggregator
[18] https://github.com/NIIF/simplesamlphp-module-attributefromrestapi

(b) a more careful approach to merging accounts, and (c) perhaps as an extension some means of re-validating attributes from time to time, to ensure they are current.

2.4.1.4 *OpenConext*

OpenConext[19] is an open source collaboration management platform. It provides a proxy for identity federation, a group proxy for group management and built-in tools for the management of services, attribute release, WAYF, metadata and access control for authentication and group providers.

OpenConext supports proxying SAML2 and OpenID Connect for authentication, XACML for fine-grained authorization and attribute aggregation by the proxy on behalf of connected services.

OpenConext is an infrastructure that enables groups, teams or organizations to bring together a set of federated tools such as wikis, mailing lists, or video conferencing software for use in a collaboration. OpenConext is being developed by SURFnet and others, and is Apache 2 licenced. The SURFnet-operated SURFconext instance currently handles about 50 million authentications per year for about 900,000 unique users.

2.4.1.5 *Perun*

Perun[20] supports various flows of user enrolment into the system. Users can be either automatically imported from an external system like LDAP or SQL database, or can go through a self-registration process where they have to fill in an enrolment form. The enrolment form is fully customizable, in terms of which attributes will be required from the users. Required attributes can be pre-filled based on values parsed from authentication request (e.g. SAML assertion or X.509 extension) and then the user may or may not be allowed to modify them, depending on defined workflow. Users can have different attributes in each VO or even have the same attributes, but with different values.

Alternatively, it is possible to have attributes on user level regardless of virtual organizations or even combine both these approaches. It all depends on concrete use-case and how the attribute workflow is defined for it. Every attribute in Perun system has its own access rights; therefore some of the attributes derived from external sources cannot be modified by the user, not even by the VO manager. Stored attributes can be aggregated and provided to services or to other systems using already described PUSH and PULL mechanisms via various protocols.

2.4.2 Use cases

2.4.2.1 *VOPaaS*

The GEANT 4 Virtual Organisation Platform as a Service (VOpaas) activity aims to deliver a set of services that support Virtual and Collaborative Organisations in their use of AAI. The activity investigated the requirements for

[19] https://www.openconext.org
[20] https://perun.cesnet.cz

such a platform in 2015, resulting in a market analysis[21]. Based on the market analysis, the service will deploy two types of services:

- **Basic services** will offer entry-level AAI services. The services will become available to all users of eduGAIN enabled Identity providers that are allowed to use this service by their Federation/NREN. Initially the basic services will include:
  - a service for onboarding VO members and providing a persistent identifier for these members;
  - a proxy for services to connect to existing Guest Identity providers solutions, including social, NREN operated and commercial solutions;
- **Advanced services** will offer per-VO services which can be selected from a portfolio. The services will be operated by GÉANT, for which a fee may be required. Services could include: attribute management tools like CoManage, HEXAA and PERUN; SP proxy tools like OpenConext; and a Step up Authentication service.

The initial services of the basic set will become available for piloting in the first-half of 2016.

2.4.2.2 *ELIXIR*

ELIXIR AAI is effectively an authentication proxy and an associated aggregate of user attributes gathered from various sources or managed locally by ELIXIR AAI and delivered to the relying services.

- ELIXIR AAI allows the user to log in using various Identity Providers external to ELIXIR (such as eduGAIN, Google, LinkedIn and ORCID) and consolidates the external identities (identifiers) to a single ELIXIR identity. Technically, ELIXIR AAI assigns two identifiers to users: ELIXIR identifier (non-revocable opaque identifier presented using the eduPersonUniqueID attribute such as: `28c5353b8bb34984a8bd4169ba94c606@elixir-europe.org`) and an ELIXIR username (a revocable non-reassignable human-readable identifier presented using the eduPersonPrincipalName attribute, for example `tommioffinland@elixir-europe.org`).
- The user's affiliation information (eduPersonAffiliation) is fetched from their Home Organisation using eduGAIN, if possible. User's authorisation to use relying services may depend on their affiliation to (or departure from) their Home Organisation.
- The user's ORCID identifier is another attribute of an ELIXIR user. It is retrieved using the APIs provided by ORCID and requires the user to authenticate at ORCID using OAuth2.
- ELIXIR AAI manages user's' group memberships (using the Perun software) and entitlements to access sensitive datasets (using the REMS software) and presents them to related relying services as attributes of the users.
- ELIXIR has defined a concept of a *bona fide researcher* with a lightweight administration procedure relying on the tools provided by the ELIXIR AAI. Some relying services are permitted only for researchers who have the *bona fide* qualification.
- Users can also self-manage some of their attributes (such as preferred e-mail address) using the UI provided by the ELIXIR AAI.

[21] http://www.geant.org/Projects/GEANT_Project_GN4-1/deliverables/D9-2_Market-Analysis-for-Virtual-Organisation-Platform-as-a-Service.pdf

The basic scenario is that ELIXIR AAI provides the necessary attributes to the relying services at the login time (on SAML or OpenID Connect). If necessary, relying services can also integrate with ELIXIR AAI using a push model where ELIXIR AAI transfers user attributes to the services in a batch file.

2.4.2.3 *EGI*

The EGI SP proxy supports attribute aggregation through:

- LDAP Attribute Aggregator SimpleSAMLphp module - Allows SimpleSAMLphp to issue LDAP queries for retrieving attributes.
- AQ Attribute Aggregator SimpleSAMLphp module - Enables SimpleSAMLphp to issue SAML 2.0 attribute queries to Attribute Authorities that support SAML 2.0 SOAP binding.
- OpenConext attribute aggregation Java application - Handles attribute aggregation and provides REST API for accessing attribute information.

2.4.2.4 *EUDAT*

The EUDAT use case can be summarised as follows:

- Credential harmonisation - ensuring credentials are consistent regardless of which attributes are supplied by the IdPs. This is done by adding attributes as needed to the internal credential (generated by the B2ACCESS proxy). However, it is still an issue that a given attribute could itself have different LoAs (Levels of Assurance), which leads to:
  - LoA of individual attributes (such as email). The provenance of an individual attribute - did it come from a trusted IdP, a less trusted one, from the user, from an administrator - leads to the requirement to track the attribute for individual attributes (in addition to having a general LoA attribute for each IdP). EUDAT is currently (as of April 2016) working on a proposal for this, based in part on AARC MNA3.1.
- There is an occasional need to rename attributes to ensure global uniqueness. Suppose an attribute “role” is published - for the same user - by two different community attribute authorities. A role of “admin” (say) might make sense within one community but is ambiguous when taken out of the community context. Thus, either one needs to rename the attribute (`communityXRole=admin`, `communityYrole=member`) or rename the value (`role=communityX.admin`, `role=communityY.admin`). At the moment EUDAT is leaning towards the latter arrangement, in order to simplify the schema used in B2ACCESS (it is easier with a multivalued ‘role’ attribute), but community attribute authorities (other than IdPs) are not integrated as of this writing.

2.5 Issues

2.5.1 Harmonization

This section has described various methods and implementations for making attributes available, particularly those from the home institute IdM. These attributes generally pertain to the user’s identity and their role within

the home organisation. As was seen previously with Shibboleth deployments (specifically the eduPersonEntitlement attribute), authorisation processes based on these attributes can be challenging: the home IdM does not necessarily know which communities the user is a member of, nor what role the user has in those communities. It thus becomes necessary to integrate community (as well as user-supplied) attributes, as provided by, respectively, a community AA (which is not necessarily itself an IdP[22]), or the users themselves.

One issues of less concern is the use of standards - all of the implementations described above are using open and free standards such as SAML and OIDC. The main issue for making attributes available is the schema used by the IdPs. Most IdPs will use schemata similar to or derived from inetOrgPerson (RFC2798), such as eduPerson, but issues then arrive when we ask whether attributes are being published consistently: what one IdP calls (say) eduPersonScopedAffiliation is the same as those published by another (the distinctions between staff, member, etc.)

However, it is easy to add custom or exotic attributes to IdPs; for example, Unity adds a `unity:persistent` attribute. It thus becomes necessary for the consumer of the attribute to make sense of it, and translate it into something meaningful. In general, this highlights the need for at least guidance in how to use attributes to promote interoperation, and possibly further harmonisation activities.

As with any reliance on external software projects, one needs to worry about the languages supported and the ongoing sustainability of the software.

[22] If the community AA is an IdP, it becomes easier if the infrastructure (and the user) permits linking the home ID with the community ID; attributes can then be combined.

2.5.2 Attribute Aggregation Models

The following issues arise with regards to the Push and Pull attribute aggregation models:

- Both PULL and PUSH
  - How does the SP discover the AAs, and decide which ones to trust? Currently models are fairly static - trusted AAs are defined by administrators - but is there a need for more dynamic models?
  - What happens if the same attribute is provided by two different AAs? What if they are used inconsistently? In general one would need to either rename them (from `role=editor` to `urn:eu:aarc:role=editor`) to ensure uniqueness, or alternatively pick the most trusted one (when receiving `mail=j.user@mailhost.com` and `mail=joe.user@organisation.ac.uk`, choose the latter.)
  - In both cases described in the preceding paragraph, the attributes - as issued by the AA to either the user or the SP - are used (for access control decisions or accounting purposes) when users authenticate (for accounting purposes, the relevant values will typically be remembered until accounting records can be made). What happens if the value of an attribute changes during the execution of a user's task? In "usage control" a PDP could be notified of an attribute changing (e.g. reputation or credits) and re-evaluate its access control decision. Of course the PEP also needs notification of a change of decision.

- PUSH
  - Users will need in general to be able to combine several sets of attributes with their own credential into one "fat" credential which can be pushed to the SP at authentication time. How is this done securely - without losing signatures from AAs (if available)? Particularly if different AAs issue attributes in different formats (perhaps one is using JSON with JWS, another uses SAML, and one uses attribute certificates)? There is currently no general solution to this problem other than to proxy or translate the attributes (thus losing the original signature), but such a scheme will not solve other issues, for example where attributes are encrypted for the ultimate recipient.

- PULL
  - Potentially each SP will need to pull attributes from each relevant AA before being able to decide which attributes to use. Thus, the scalability issue may lie with the number of SPs, the number of AAs, the number of users, or the number of connections (or all of them): Each SP will need for each user, for each authentication (or each access control decision), for each relevant AA, to pull an attribute set, limited only by caching on the SP's side.
  - How does the AA know that it can trust the SP and release attributes to it? Like 1a, it is currently statically configured (i.e. by an administrator) but more dynamic models could be useful.
  - And the flip side of 3.b, how does the SP authenticate to the AA and obtain attributes for the user in question? Specifically, how can the SP be restricted to only obtain attributes for the user and not some other person? OAuth2/OIDC can restrict access, depending on implementation, but there is no general solution combining user consent with authorisation for remote parties to access user attributes.

2.5.3 Consent

One of the specific problems with publishing attributes is whether users have consented to their data being published. Data protection becomes easier when user consent can be sought and recorded, and in practice consent must always be sought at some level:

- By using the service, users consent to their attributes being published (implicit).
- Users give their consent every time an attribute is released (explicit).

It helps, of course, if the user knows how the attribute will be processed. Few users, perhaps, would read a document but public services in general have a privacy statement somewhere, which explains how user data is processed.

2.5.4 Attribute's Level of Assurance

Until now, the concept of Level of Assurance (LoA) has been applied to authentication and identity proofing, and different Levels of Assurance, mainly based on the NIST Electronic Authentication Guideline[23] and the ISO/IEC 29115:2013 Standard[24], has been successfully deployed in National Identity Federations. The strongest Level of Assurances correspond with the strongest means to proof the identity being authenticated, and more robust vetting procedures to be employed in the enrolling phase. LoAs are extremely useful as a mean to control who can access sensitive or critical data assets and services, nonetheless they are limited to the identity profiling side of the authentication and authorization process. In an integrated AAI environment, where exchange of attributes among different entities is a key aspect, LoAs are of little use to provide trust levels for individual attributes.

The current lack of a trust model to convey assurance information about individual attributes can be overcome using the same techniques employed by LoA, namely assertions, but a real standard has not yet emerged.

In cases where Attribute Aggregation processes are being used the challenge becomes bigger. In fact, when Attribute Aggregation is being employed, the entity that transmits the attributes could be different from the one that issued and released them. When released attributes has attached statements about the Level of Assurance, the entity that will receive the assertions has not only to trust the entity that performed attributes aggregation of not having altered the content of the third-parties attributes, but also of not having in any way added or altered assurance information.

[23] Electronic Authentication Guideline, Special Publication 800-63-2, NIST, 2003, http://csrc.nist.gov/publications/nistpubs/800-63-1/SP-800-63-1.pdf
[24] ISO/IEC 29115:2013, ISO, 2013.

3 Attribute Translation

3.1 Group membership to authorisation

Both groups and attributes can be used as the basis for authorisation. Which authorisation method should be used? The advantages and disadvantages of the two methods are described below:

Attributes:

- *Advantages of Attributes*
  - Agreements between the institution and the provider on the value of an attribute for cloud and other services are easy to make. The attributes concerned (entitlements) are also relatively easy to manage in the institution's own identity management system.
  - The attributes are available as soon as users log in, so that the entitlement can be used for screening at the gate (coarse-grained authorisation).

- *Disadvantages of Attributes*
  - If there are many small groups of users, each needing its own specific entitlement for services, this can generate extra work for the administrator of the institution's identity management system.
  - In case of supra-institutional collaboration, the parties have to make clear agreements on attribute naming, as the administrators of other identity management systems must allocate the same attribute.

Groups:

- *Advantages of Groups*
  - Groups can be very diverse. In terms of scale, they work with both a very small number of users and a very large number of users per group.
  - The composition of the group is normally regulated in one place, which makes it easier to re-use groups for several services.
  - Group membership is requested by the application, after the user has already received access to the application, which makes it possible to screen and/or share rights on the basis of this group membership within the application.
  - It is also possible to request information within the application when the user is not currently logged in.

- *Disadvantages of Groups*
  - An institution must create a separate 'group provider' (with the LDAP/AD as source to re-use the institution's existing group relationships).
  - A service can potentially see all group memberships of a user per group provider and per service combination. When a service has access to the group provider, it can see all the relevant provider's groups, though it is possible to display only the group names and not the members.

- Finally, the service must be suited to request the information from the Group provider API.

Combining both methods is also possible, with entitlements being used for access at the gate and group memberships for rights within the application.

An example of this is to use the VOOT plugin in the SimpleSAMLphp software for service providers. Upon logging in, the plugin reads the user's group memberships and transmits them to the application together with the SAML attributes. The advantage of this is that the service does not need to make any modifications to request the API, but the group memberships can only be requested upon login.

3.2 SAML to OIDC claims mapping

The REFEDS OpenID Connect for Research and Education Working Group (https://wiki.refeds.org/x/TYMY) is working to standardize the mapping from SAML attributes (eduPerson and SCHAC) to OIDC claims, for submission to the IANA JSON Web Token Claims Registry. Currently, services like CILogon (http://www.cilogon.org/oidc) and AAF Rapid Connect (https://rapid.aaf.edu.au/developers) perform this mapping using different, locally defined OIDC claim names.

OIDC supports two types of subject (sub) identifiers, public and pairwise, which map well to eduPersonUniqueId (ePUID) and eduPersonTargetedID (ePTID) (SAML2 Persistent NameID), respectively. The OIDC subject identifier must not be re-assigned, making eduPersonPrincipalName (ePPN) a poor candidate for mapping from SAML to OIDC.

The remainder of this subsection describes how some of the most commonly used social identity profiles can be mapped to SAML attribute assertions. It should be noted that only user information which is relevant to the REFEDS Research and Scholarship (R&S[25]) attribute bundle is covered here.

3.2.1 Google (OpenID Connect)

Google's OAuth 2.0 APIs can be used for both authentication and authorisation. This OAuth 2.0 implementation conforms to the OIDC specification and is OpenID Certified. As such, when including the openid scope, information about the user can be retrieved from the UserInfo endpoint in OpenID Connect format. The Claims[26] returned in the UserInfo Response can be mapped to SAML attributes as shown in Table 1. Note that the included sub is the user identifier, which is unique among all Google accounts, persistent and non-reassignable. When mapping the sub to an ePUID this needs to be scoped at "google.com", to adhere to the eduPerson specification.

3.2.2 Facebook

Facebook allows the retrieval of user information through the `/{user-id}` Graph API endpoint[27], following an OAuth 2.0 flow for authentication and authorisation. The returned fields of the Facebook user profile can be

[25] https://refeds.org/category/research-and-scholarship
[26] https://openid.net/specs/openid-connect-basic-1_0.html#StandardClaims
[27] https://developers.facebook.com/docs/graph-api/reference/v2.6/user

mapped to SAML attributes as per Table 1. Note that the Graph API returns by default only the ID, i.e. a user ID unique to each app that cannot be used across different apps. Effectively, this is a targeted ID that can be mapped to an ePTID. In order to retrieve an anonymous, but unique identifier for the user that can be shared with third-parties, the `third_party_id` field must be requested explicitly. Conceptually, `third_party_id` should be mapped to an ePUID[28] which is also opaque (and non-targeted). However, a `third_party_id` may contain non-alphanumeric characters which are not valid for the uid portion of an ePUID. Thus, `third_party_id` is mapped to an ePPN (scoped at "facebook.com"), which is not required to be human-friendly.

3.2.3 LinkedIn

LinkedIn relies on the OAuth 2.0 protocol for enabling authenticated access to its REST APIs that provide access to member data. More specifically, following a three-legged OAuth2 flow, LinkedIn user profile[29] information can be accessed through the `/people/~` REST API endpoint. Table 1 shows how the returned user fields can be mapped to SAML attributes. Note that the ID user field returned by LinkedIn is a unique identifying value for the user, which is firmly associated with the specific application; thus, it should be mapped to an ePTID.

[28] http://software.internet2.edu/eduperson/internet2-mace-dir-eduperson-201602.html#eduPersonUniqueId
[29] https://developer.linkedin.com/docs/fields/basic-profile

SAML	Google (OIDC)	Facebook	Linkedin
ePUID	sub	n/a	n/a
ePPN	n/a	third_party_id	n/a
ePTID	n/a	id	id
displayName	name	name	formatted-name
givenName	given_name	first_name	first-name
sn	family_name	last_name	last-name
mail	email	email	email-address

Table 3.1: SAML-to-Social user attribute mappings

3.3 SAML to x509

As identified in section 2.1, one of the common issues is the choice of schema in publishing attributes. It is quite common to base user identity attributes on schemata derived from inetOrgPerson (RFC2798). In general with LDAP or X.509, the user identify is formatted as a hierarchical Distinguished Name (DN), based on existing standards (such as X.500 and RFC5280) and where the international community has derived, through years of practical experiences, guidance for using them properly (such as GFD.225[30]). For both X.509 and LDAP there must exist an encoding of the attribute into ASN.1; in other words, each attribute must have a well defined OID. Conversely, experiences have shown that it helps if this OID, in turn, is known to commonly used tools like OpenSSL and JKS, *and the attribute name is used consistently across all such implementations*. The classic example is the attribute `emailAddress` which has also been rendered (incorrectly) in string form as email or E, thus leading to interoperation issues; see GFD.225 for discussion. Other issues arise with the encoding: for example, a principal would contain '@' which cannot be encoded as `printableString`; again, refer to GFD.225 for a full discussion.

SAML, in turn, carries an assertion on the subject, which could be used consistently with X.509 in order to promote interoperation, in other words make the SAML subject the DN. As SAML is XML, the encoding issues are less of a concern.

3.4 SAML-to-SAML

SAML-to-SAML translation is very common in AAI architectures that include Proxy components. Typically, the Proxy components in the implementation "consume" SAML requests and assertions from SPs and IdPs and "produce" translated SAML requests and assertions, which are relayed to upstream IdPs or to downstream services.

[30] https://redmine.ogf.org/dmsf_files/25?download=, see also https://www.ogf.org/documents/GFD.125.pdf

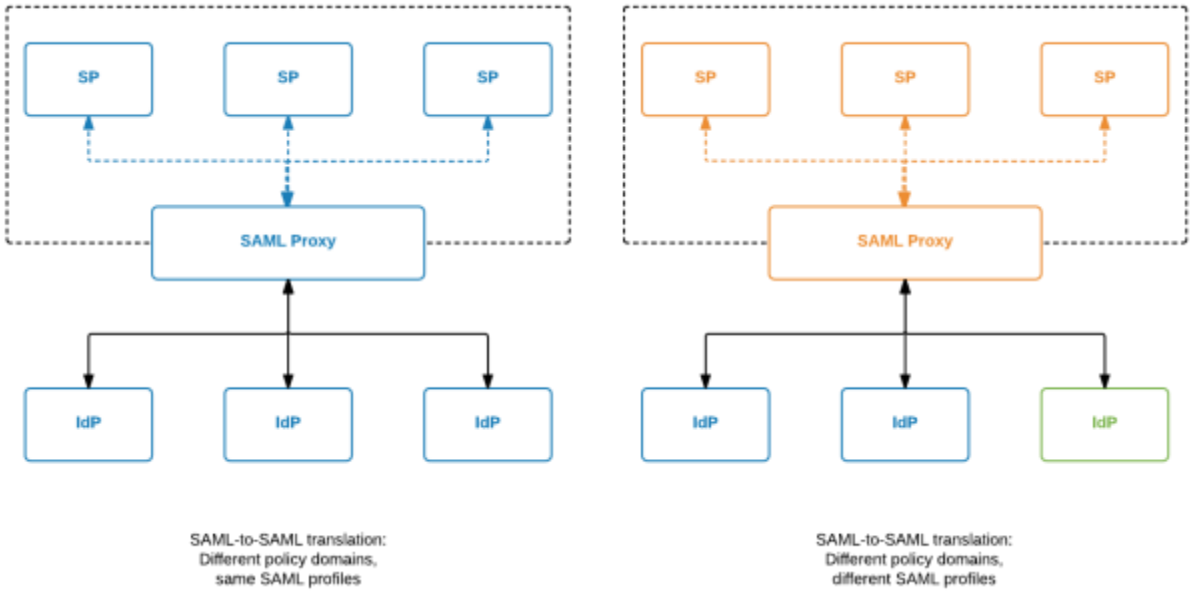

Figure 3.1: SAML-to-SAML translation

In implementations, which are using the same SAML profiles, the translation is simpler. Basically, the operation of the Proxy revolves around decrypting, verifying and parsing the contents of the original SAML message in order to compose, sign and encrypt a new SAML message using new key pairs, and replacing parts or all of the information of the entity that created the original SAML message with those of the Proxy component. The actual contents of the SAML message might be enriched or modified, depending on the specific use cases.

A typical scenario of enriching the original SAML message is when the Proxying component receives a SAML assertion from an IdP and extracts the information related to the user identity. This information is then encapsulated in a new SAML assertion generated by the Proxy components, which includes more information about the user than might be available to the Proxy component via other means. Another typical scenario is when the Proxy components receives a SAML assertion message by an IdP, which include IdP specific attributes that need to be translated to attributes that the downstream services behind the IdP expect. For example, an IdP might be releasing entitlement attributes in a specific format, which the Proxy component has to translate to a format that is recognized by the connected downstream services.

A more complex case of SAML-to-SAML translation can be found when a Proxy component is meant to act as a bridge or a gateway between services that are using different SAML profiles. For example, the R&E Federations rely on the standardized SAML2Int profile in order to ensure interoperation between SPs and IdPs within and across Federations. The eIDAS Interoperability Framework, on the other hand, which is deployed in Europe in order to enable cross-border interoperation of eGov IDs, relies on a proprietary SAML profile, with non-standard SAML extensions. In order to enable interoperability between services in the existing R&E federations and eIDAS, proxy components are required, which will translate SAML messages following the SAML2Int profile to assertions and requests compatible with the SAML profile used in eIDAS. This SAML-to-SAML translation is significantly more complicated, as the component responsible for the translation has to be able to translate between different attributes, which in many cases do not have a one-to-one mapping.

More information about the differences between the R&E Federations and eIDAS and the challenges in building interoperable solutions can be found in the report that was produced by GN3plus project in 2014[31]. This analysed the different architectures and implementations between the federations in the R&E, and specifically the production federations, which were already members of eduGAIN, and the STORK2 architecture and implementation, which was meant to be the reference for the eIDAS deployment.

[31] The work has been presented at TNC15 Networking Conference, https://tnc15.terena.org/getfile/2053

4 Guidelines and recommendations

4.1 Improving attribute release

Attribute release is a crucial and complex issue for Identity Federations. Usually, the attribute release policy employed by Identity Providers is strongly focused on data protection issues; this is perfectly reasonable since IdPs are responsible for the information they release about their users. Apart from authentication assertions, information covered by the data protection laws like Personal data, or Personally Identifiable Information (PII), are released only when a trust channel (or a specific contract) is established between the IdP and the Service Provider that will consume the information. This process does not scale well, even for a National Identity Federation, but its shortcomings are paramount when inter-federations like eduGAIN are concerned. The end result is a lack of attribute release that poses strong limitations on the scientific community and on international cooperation.

4.1.1 The REFEDS Research and Scholarship Entity Category

The REFEDS Research and Scholarship entity category[32] was designed to provide a safe way to release a small attribute bundle to Service Providers that have proven they require the attributes requested for research and scholarship (R&S) needs.

Registration in the R&S entity category is up to the Home Federation of the Service Provider. The Registrar, among other technical requirements, must assess that the service enhances the research and scholarship activities of some subset of the registrar's user community.

4.1.2 How to deal with missing attributes on the SP side

Service Providers that cannot give access to resources due to a lack of attribute release can and should employ some strategies to ease the process of attribute releasing, in other words: to fail, but fail gracefully.

A key point is to inform the user about the issue: usually, when there is a lack of attribute release the user cannot access the service, so the information about the real cause of the problem is poor or non-existant. The result is that the user is discouraged from accessing the service and the lack of attribute releasing is never solved, since a user will never ask their IdP administrators to craft a suitable access policy for that particular SP.

Improving the information given to users about the failure, and pointing them to their Identity Providers administrators to overcome the issue can lead to a virtuous circle, where the issues that prevent the use of the service (the lack of attributes release) become apparent, and the people that can help in solving the issues (the IdP administrators) are clearly identified.

While this process is not hard to implement by Service Providers at the application level, it can pose a burden on others. For those SPs that are short of development resources, or that do not want to or cannot easily implement

[32] https://refeds.org/category/research-and-scholarship

the “graceful fail” at the application level, there is the option to implement it directly at the SP level by leveraging the Attribute Checker hook in the Shibboleth Service Provider. Once configured, the AttributeChecker validates the user session against attributes specified as required. If these are fulfilled, the login completes, otherwise an error page is displayed.

Detailed instructions on how to configure the Attribute Checker hook, and how to customize the error page to report released attributes, missing ones, IdP contact information (extracted from the metadata), and even a template for reporting the issue to IdP administrators can be found on the eduGAIN wiki[33].

4.2 Future work

This Milestone includes an analysis and comparison of many different aspects of an integrated and interoperable AAI. Important topics left out, like Token Translation Services, Non Web-Access, and Delegation of authorisation, will be addressed in AARC MJRA1.4 ‘First Draft of the blueprint architecture’.

[33] https://wiki.edugain.org/How_to_configure_Shibboleth_SP_attribute_checker

Glossary

AA	Attribute Authority
AAI	Authentication and Authorisation Infrastructure
AARC	Authentication and Authorisation for Research and Collaboration
API	Application Program Interface
CA	Certificate Authority
COU	Collaborative Organizational Unit
eduGAIN	EDUcation Global Authentication INfrastructure
ePUID	eduPersonUniqueID
ePTID	eduPersonTransientID
ePPN	eduPersonPrincipalName
EUDAT	European Association of Databases for Education and Training
GUI	Graphical User Interface
ID	Identity
IdP	Identity Provider
JRA	Joint Research Activity
JRA1.4	AARC JRA1 Task 4 Models for implementing Attribute Providers and Token Translation Services
LoA	Level of Assurance
NIST	National Institute of Standards and Technology
NREN	National Research and Education Network
OIDC	OpenID Connect
PEP	Policy Enforcement Point
PIP	Policy Information Point
R&E	Research and Education
R&S	Research and Scholarship
REFEDS	Research and Education FEDerations group
RP	Relying Party
SAML	Security Assertion Markup Language
SCIM	System for Cross-domain Identity Management
SOAP	Simple Object Access Protocol
SP	Service Provider
SSO	Single Sign On
STS	Security Token Service
VOMS	Virtual Organization Membership Service
VOOT	Virtual Organisation Orthogonal Technology
WebSSO	Web Single Sign On
XACML	eXtensible Access Control Markup Language