Implementing an Authorisation Architecture in the EUDAT Services Federation

Shiraz Memon∗§, Jens Jensen†, Willem Elbers‡, Morris Riedel∗§, Helmut Neukirchen§, Matthias Book§

∗Jülich Supercomputing Center, Jülich Germany
{a.memon, m.riedel}@fz-juelich.de
†CLARIN ERIC, Utrecht, Netherlands
willem@clarin.eu
‡STFC, Oxford, United Kingdom
jens.jensen@stfc.ac.uk
§University of Iceland, Reykjavik, Iceland
{asm25, morris, helmut, book}@hi.is

Abstract—This paper describes the requirements and architecture of authorisation in a multi-disciplinary, multi-site, multi-stakeholder infrastructure which is using federated identity management. Stakeholders include administrators of sites, infrastructure, services, as well as data owners and community representatives. In order to be able to express and combine policies, we have based the authorisation infrastructure on XACML.

Index Terms—Federated Authorisation, XACML, AAI, e-infrastructure, cyberinfrastructure

I. INTRODUCTION

Authorisation is the decision that is taken by a service when a user wishes to perform an action on the service. In its simplest form[1], it has a user accessing a service based on the user’s identity or rights granted by an authority (the home organisation in the RFC). In more complicated cases, rights can be delegated, for example from the user to a client acting on behalf of the user, or authorisation may be fine-grained and depend on the type of action the client wishes to perform, and on which object.

In this paper, we focus on an e-infrastructure/cyberinfrastructure (section II-A) that has already implemented Federated Identity Management (FIM). It is multi-disciplinary, so users may belong to more than one user community, or may be collaborating across communities, so the authorisation system must meet the requirements (section III) of user communities, home organisations, service providers, data owners, as well as the infrastructure operations. In addition, an authorisation service must of course be resilient, have sufficient performance, be sufficiently fine-grained to meet the requirements, and be sufficiently expressive and usable that it is actually used to implement the required protection policies. Further merits include being based on open standards, and having multiple interoperable implementations.

The novelty of the work presented here is precisely this balancing act: we needed a service that interfaces with the existing infrastructure, yet meets the needs of all the stakeholders mentioned earlier. Some of the constraints are technical (protocol, connecting to extant attribute providers), some are a question of software engineering (service integration, tests) and infrastructure operations (deployment time-scale), some are policy-based (implementing and combining data and service policies), some are architectural (section IV) and some are a question of “inertia” (for example interfacing with existing practices in communities.)

II. BACKGROUND

A. EUDAT

European Data Infrastructure (EUDAT)[2] is an e/cyberinfrastructure that provides data storage and management for a wide range of research areas[1]. The project grants access to its services based on a FIM service called B2ACCESS[3], which accepts several types of identity providers (IdPs) that are external to the project, and produces a harmonised authentication credential which is consumed by all services in the infrastructure.

EUDAT comprises several data “B2” services (B2SHARE, B2DROP, B2FIND, etc.) for the user communities, plus “internal” services that support the infrastructure itself (Wiki, JIRA, helpdesk, etc.). Nearly all of these services require authentication (through B2ACCESS). B2ACCESS provides user provisioning, attribute management, credential translation, trust management, and entitlements management, as well as the authorisation which is the topic of this paper.

B. Challenges

The main concern of authorisation in EUDAT is to manage the end user’s (write) access to shared resources, as well as (read) access to data and meta-data, based on a combination of policies from different stakeholders. In particular, there is a set of infrastructure-specific and community-specific roles that need to be managed, including the rights to grant the roles to others. The community-specific roles are typically managed

1To be precise, EUDAT is both a project, EUDAT2020, funded by the European Commission as a part of the Horizon2020 programme, and a sustainable “collaborative data infrastructure” run by a group of organisations, which includes the project partners.
by people authorised by the communities, that is, they may be maintained on disparate services outside of EUDAT and must be imported. At the service end, EUDAT provides a range of data services which in turn are RFC-developed on top of diverse components developed outside of the EUDAT project.

C1 Distributed authorisation. The first of the main challenges for EUDAT is thus to implement authorisation in a way that meets the requirements of the user communities, and is technically implementable across the different technologies used in the infrastructure, in particular when a FIM model is used for authentication. It must also be trustworthy and consistent across services, in order that data owners feel that their data is adequately protected.

C2 Harmonisation. EUDAT also needs to function as an infrastructure, so authorisation information from the communities may need to be harmonised in order to be enforceable consistently across the infrastructure. The second main challenge (C2) is then to make the authorisation implementation consistent across all services and across all service providers in the infrastructure, while also making it scalable in the number of objects it protects and the frequency of actions, and to make it resilient against intermittent network failures.

C3 Usability. The third main challenge is that the authorisation services should be usable. If people do not make use of the features because they are complicated or hard to manage, then the service will not be useful. EUDAT’s authentication system takes the simple “portal” approach by default, while providing more complex approaches (such as command line) as options for expert users, and it is likely that the same approach will work for authorisation. A related, but different, aspect of this challenge is the expressiveness of the policy language: If it is hard for users to express their policy in the language, they will work around the authorisation system, or will go elsewhere.

C4 standards and interoperability. The fourth and final challenge is to be reasonably future-proof: the authorisation subsystem should be based on mature and open international standards and be interoperable across multiple implementations (particularly in order to support services implemented in different programming languages.) In particular, this approach should foster harmonisation and best practices between EUDAT and peer infrastructures.

In this paper, our focus is primarily on C1 and C4; we will not have space here to deal with all and, in fact, not all have been fully addressed yet in the project.

C. Context and Related Work

At a high level, the context of the work presented here is the need for “federated authorisation” to support e-cyberinfrastructures as summarised by [4] and [5]. In particular, the context includes authorisation as implemented by our peer infrastructures [5], and for both these and EUDAT, the use of FIM needs to be followed by an authorisation model which fits with the established authentication methods.

The original concept of Grid computing included Virtual Organisations (VOs) as a means of managing communities, and, as in EUDAT, there was a need to manage policies from multiple stakeholders and resource/quotas in a suitably expressive and flexible way [6]. Authorisation was split into the several parts:

1) The VO negotiates with sites to obtain resources for its members and defines the scope of the work, etc. Roles and their rights are defined.
2) Individual users, once they can authenticate, request membership of the VO.
3) Authorisation attributes are assigned to individual users (by their principal)
4) Users request roles (and, possibly other authorisation attributes).
5) When users access resources, they are typically mapped to a local user id based on their role, or if they have no role, a default id assigned to the VO members.

As we shall see (section IV-C), we chose to use eXtensible Access Control Markup Language (XACML) for our implementation. The most important prior work of direct relevance to EUDAT is [7].

Outside of the science world, authorisation in distributed environments have been studied as well.

III. REQUIREMENTS

Given the distributed nature of the EUDAT infrastructure, we have identified the following high-level requirements:

R1 : Easy, centralised and delegated management of authorisation policies.
R2 : A resilient, scalable, and highly available authorisation infrastructure.
R3 : Auditable authorisation policies.

Addressing challenges C1-C3, R1 requires there to be an architecturally central service [8] to manage authorisation on behalf of all the stakeholders. R2 is a standard requirement and not specific to our project; and, addresses C2: how to ensure that the implementation of the policies is correct and consistent across the infrastructure.

A. Use Case

We use the EUDAT B2SAFE and B2STAGE services as an example use case to illustrate two of the requirements. B2SAFE provides safe (i.e. replicated) storage across several data centres, and because access is enforced at the service level, there is a need for harmonised authorisation [R1]. B2STAGE is used to move data between EUDAT and other infrastructures, as illustrated in Fig. 1 and introduces the requirements [R1] and [R2].

A typical workflow is a community user running a simulation and then importing the results into B2SAFE by using the B2STAGE API, making multiple copies in different locations. For this to work, we need to be able to allow write access to specific storage resources in different data centers, based on the user’s attributes (such as community membership and role).
Since multiple data centers need to have access to the authorisation policies in real time, there is a clear need for a scalable and highly available solution (R2). If there is any issue in the communication between EUDAT centers, each B2SAFE should still be able to make authorisation decisions. This demonstrates the need for a distributed authorisation infrastructure (even if, architecturally, it is a single, central, service: There is only one authorisation service, but it must offer multiple service endpoints).

**B. RBAC vs ABAC**

Attributes are associated with user identities in the EUDAT infrastructure. Therefore, an Attribute-based Access Control (ABAC) approach is preferred over a more traditional Role-based Access Control (RBAC). ABAC provides more flexibility and finer granularity over the more traditional RBAC approach: RBAC requires defining the roles upfront, whereas ABAC requires only the upfront definition of a set of attributes. The ABAC approach matches with the set-up of B2ACCESS, where a fixed set of attributes is already defined and associated with each user. Moreover, as described in [9], an ABAC-based approach does not exclude roles. In ABAC, attributes with role names can be introduced together with rules controlling the modes of access to the protected objects. One of the disadvantages of ABAC, as mentioned in [9], is that auditing (III) is more difficult because the set of attributes/values is dynamic, making it more difficult to enumerate all possibilities. Within B2ACCESS, the set of attributes is fixed, and because of the proxy-like nature of B2ACCESS[8], attribute values are cached at the B2ACCESS service, making it possible to make a snapshot of all identities with access to the EUDAT infrastructure and their associated set of attributes and values. The set of authorisation policies is also centrally available, making it relatively easy to compute the set of permissions of a user at a given moment in time, allowing us to fulfil (RIII). For example, the B2SHARE service requires following (more than merely role) attributes to grant sharing or upload access rights to a user: community-name (subject is associated with), community-role (the subject has within the community), email (to receive/send sharing requests and notifications), user workspace, endpoint URI (resource information) and share / upload data (the invoked action).

**IV. ARCHITECTURE**

Based on work in existing infrastructures – including EU-DAT – the AARC project identified a common architecture for authentication and authorisation[8].

**A. Authentication and user management**

The EUDAT authentication service, B2ACCESS, enables users to authenticate, and provides account management. Its features include:

1) support for multiple external authentication protocols (OpenID Connect (OIDC), SAML, X.509, LDAP), and translation of security tokens between different authentication protocols
2) integration with eduGAIN[10], thus supporting identities from hundreds of Universities and Research institutions around the world
3) provisioning of a single user account, and a unique representation of the user identity to the infrastructure
4) user account de-provisioning (i.e. users can request to be "forgotten")
5) support for the proxy Identity Provider (IdP)/Service Provider (SP) concept[8] (acting as an SP to external IdPs and as an IdP to the SPs, i.e. the EUDAT services)
6) "enrichment" of user identities with extra infrastructure-specific attributes (cf. III-B)
7) management of users and attributes in groups, representing user communities (e.g. CLARIN[11], EPOS[12], ENES[13])

**B. Authorisation Model**

Authorisation in EUDAT is based on OASIS’s XACML[14], which specifies an access control policy language instead of the more traditional Access Control Lists (ACLs). XACML implicitly supports (see IV-C) Attribute-Based Access Control (ABAC), and relies on the evaluation of subject, resource, and environment attributes to form an access decision.
In addition, XACML provides an abstract architecture, which consists of the following five components (see Figure 2).

1) **Policy Administration Point (PAP):** The PAP is an administrative service that allows stakeholders to create, manage, debug and store the relevant access control policies. Depending on the reference implementation, the authorisation services can offer a Graphical User Interface (GUI) and/or RESTful Application Programming Interface (API) to administer the policies. The current XACML standard and its profiles do not have a standardised PAP API yet.

2) **Policy Decision Point (PDP):** Also called Context handler, the Policy Decision Point (PDP) component evaluates access requests (which may contain authorisation attributes) against access control policies and computes a response, i.e. an access decision. Usually the response is **PERMIT** or **DENY**, but it can also decline to take a decision, for instance deferring the decision to another service.

3) **Policy Enforcement Point (PEP):** Usually, the PEP intercepts the access request, sends a request to PDP and acts upon the response.

4) **Policy Information Point (PIP):** The PIP is an optional component that is responsible for fetching subject attributes from external attribute providers. The authorisation service can leverage the Policy Repository (PIP) if the Policy Enforcement Point (PEP) does not submit all required attributes with the access decision request.

5) **Policy Repository:** An infrastructure needs a service which stores the policies, where they can be accessed by the PDP and the Policy Administration Point (PAP).

In addition to the policy language, XACML defines the structure of access requests and responses. For XML implementations, a normative schema XML 2 facilitates standards compliance and interoperability between implementations. The specification has also defined a Java-script Object Notation (JSON) rendering, which is only limited to request/response messages.

In addition to this, XACML defines a number of profiles for communication and integration between services, namely:

- **P1 The Administration and delegation profile** is used to express administration and delegation policies which enable administrators to delegate – and limit – administrative rights to local administrators to enforce access control on a subset of protected resources (cf. P1).
- **P2 The Security Assertion Markup Language (SAML) profile** enables integration of SAMLv2 with XACML. The PDP can consume SAML attribute assertions in order to make authorisation decisions.
- **P3 The REST profile** partially defines a RESTful API which currently focuses on communication (see P2) between the PEP and PDP.
- **P4 The Multiple decision profile** allows a requester—to send several access decision requests in one go, to which the PDP returns one answer with multiple decisions.
- **P5 The Digital signature profile** defines the authenticity and integrity of XACML schema instances using the W3C XML-Signature Syntax and Processing standard.
- **P6 The Hierarchical resource profile** provides access control for resources organised as a hierarchy, such as file systems, XML documents, or organisations.
- **P7 The Hierarchical Role-Based Access Control (RBAC) profile** defines the requirements for core and hierarchical RBAC through XACML policy language.
- **P8 Intellectual property control profile:** This profile enables service providers to write and enforce policies for the purpose of providing access control for resources deemed intellectual property.
- **P9 Privacy policy profile:** This profile lets service providers express privacy policies in XACML, which defines the limits, quality, purpose, and accountability principles of user’s personal data.

C. **The choice of XACML as the policy language**

We have briefly discussed XACML in section (IV-C). There are currently two versions v2.0 and v3.0: we mainly focus the latter as it comes with support for all the profiles of the former (P2, P5, P6, P9) and includes a new set of profiles (P1, P3, P4) and (P8). As some of the B2 services in EUDAT use OIDC for authentication (see [IV-A]), These services will need to use the JSON rendering and REST profile to communicate between the service’s PEP and the PDP. The profiles in XACML v3 will thus enable us to integrate these services.

However, EUDAT takes into account certain recommendations from earlier v2 based work, in order to promote interoperability within the infrastructure, and, eventually, across infrastructures.

- subject names are always X.509 distinguished names as in the SAML assertions (section [IV-A]), irrespective of whether users have a certificate issued to them through the X.509 “gateway”.
- attributes are fully qualified, and the PDP matches against the full attribute string
- future extensions will need to look at the obligations, where the PEP specifies which types it is prepared to honour, as they will be important for some user communities
- users may, but need not, have an X.509 certificate;
- they may, but need not, have VOMS assertions assigned to their subject name (by an authority outside of EUDAT);

Ultimately, it’s a question of usability, as the obligations can help communicate additional constraints. However, this use case is beyond the scope of the present paper.
• Users may be members of more than one community ("VO" in [1]) and will need to simultaneously assert membership of both/all, as well as roles in each one.

Policies are defined by the stakeholders. Obligations are defined by them and it is up to the PDP to send it and to the PEP to implement it.

The EUDAT operations team introduced service-specific attributes for each of the services. Requirement [R1] enables administrators to give, say, B2SHARE-specific attributes to users. With time, the associated service-specific authorisation policies will become more sophisticated, and will need maintaining by multiple parties. At the same time, each centre will define policies for all its own services. Thus, there is a need to combine policies defined by different stakeholders into policy sets applicable to the request, with appropriate combination algorithms. Although we are not using it yet, we expect the delegated administration profile [16] will make this process easier.

D. Authorisation in EUDAT

Figure 3 depicts the XACML-based hierarchical architecture, which aims to address the requirements of implementing consistent [R1] yet highly available [R2] authorisation in a distributed infrastructure.

From the top of the component hierarchy, EUDAT Authentication and Authorisation Infrastructure (AAI) consists of a central PAP and Policy Repository (PRP), the latter being a database of rules. At this level, rules are defined as Policy Sets for each type of services (section IV-C). The service-specific policies are managed by service administrators through the central PAP service. Delegation of policy administration rights will use the XACML v3.0 Administration and Delegation profile to define the policies for access to the resources. The changes made at the top-level PAP update the policies at the top-level PRP, a database of policies.

Eventually, delegation of administrative rights should encompass all policy stakeholders: data owners, community admins, resource admins, site admins, and the infrastructure admins themselves. The combination of policies needs to resolve based on the target: site admins will have priority for services at their site, community admins are authoritative only for their own data and their own users, etc.

For each EUDAT data center, there should be a full XACML stack with a PEP for each service (or a group of closely co-located services), and a single PDP for the center together with a local, read-only PRP. Although the PIP is displayed in Fig. 3, all the required information (attributes sent by B2ACCESS) is in practice sent via the PEP to the PDP.

Administrator creates or updates policy through the central (read-write) PAP (Fig. 3). The central PRP pushes these policies or policy sets to the site PRP. Each site PRP receives the update and through an eventually consistent [23] policy database updates its information. The PDP accesses only the relevant policies from the site PRP in order to evaluate the access decision requests, e.g. for a B2SHARE PEP, it will request only B2SHARE policy sets.

In future work we will introduce a message broker between the site and the central PRP to ensure the reliability of the updates.

E. Access control flow

We now revisit the use case from section III-A but first from a generic service point of view. Fig. 4 shows the generic authorisation flow for a user accessing a B2-service. The EUDAT community user (e.g. CLARIN, EPOS, ENES) initiates the authorisation flow by trying to execute a privileged action on the B2-service. This, however, requires appropriate rights on the service. Since the user is not authenticated yet, they will be redirected to the B2ACCESS service for authentication (section IV-A).
After successful FIM authentication to B2ACCESS, B2ACCESS returns an authentication token along with authorisation attributes to the B2-service, which will then retry the privileged operation.

During the retry operation, the service PEP sends an authorisation request to the (site-local) PDP service, which contains the user attributes, action and resource information. The PDP service evaluates the request (and attributes within) against the policies stored in the PRP and returns the decision which is then enforced by the PEP.

Returning to the use case (Fig. 1), the user wishes to import result data from their simulation into EUDAT via the B2STAGE service, which in turn ensures replication of data across multiple B2SAFE instances. Both B2STAGE and B2SAFE check authorisation.

Traditionally, users access the B2STAGE service through a HTTP API using command line clients, with a delegated X.509 certificate. The certificate in the current implementation always contains authorisation attributes in SAML format in a custom extension[27]. The B2STAGE can thus extract the authorisation attributes directly after successful authentication.

Assuming the user is authorised by B2STAGE based on the attributes, the service obtains a delegated certificate from the certificate the user client used to authenticate[30], which in turn contains the certificate with authorisation data. Data is copied through B2STAGE to the B2SAFE instances using GridFTP (cite GridFTP), and the B2SAFE services in turn perform their own authorisation check.

As with B2STAGE, the authentication subsystem of B2SAFE extracts the SAML assertion from the relevant certificate(s), and builds a PDP request for the requested action (data ingest) to grant/deny the access.

V. COMPARISON OF IMPLEMENTATIONS

As discussed briefly about well-known XACML implementations in Sect. IV-C, this section provides a high-level analysis of those implementations. It skips those without support for a remote PDP interface. Taking the authorisation requirements of EUDAT into account, the profiles required by the infrastructure services are Core, REST and JSON. XACML version 3.0 seemingly is the adequate standard, as the later versions are Core, REST and JSON. XACML version 3.0 seemingly is the adequate standard, as it only involves infrastructure administrators as policy managers (no delegation), and authorisation is not yet integrated with all B2 services. The current deployment is simultaneously a feasibility study, a partial implementation, and an indicator of future directions. Apart from the obvious ones, of wider deployment in more production services, future directions also include:

- Implementing the delegated administrative rights, in order to support the multi-stakeholder management of policies;
- Interoperation with other infrastructures, notably EGI, which uses XACML version 2, and PRACE which uses an LDAP based system.
- Implementation of obligations.

ACKNOWLEDGEMENT

EUDAT2020 is funded by the EU Framework H2020 – DG CONNECT e-Infrastructures, contract no. 654065.

REFERENCES

<table>
<thead>
<tr>
<th>Name</th>
<th>Spec. Version</th>
<th>Supported Profiles</th>
<th>Language</th>
<th>License</th>
<th>UI</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT&amp;T XACML</td>
<td>v2.0, v3.0</td>
<td>Core, Multiple Decision, JSON, REST</td>
<td>Java</td>
<td>Apache 2.0</td>
<td>✓</td>
</tr>
<tr>
<td>ndg-XACML</td>
<td>v2.0</td>
<td>Core, SAML 2.0</td>
<td>Python</td>
<td>BSD</td>
<td>✓</td>
</tr>
<tr>
<td>ARGUS</td>
<td>v2.0</td>
<td>Core, SAML 2.0</td>
<td>Java &amp; C</td>
<td>Apache 2.0</td>
<td>✓</td>
</tr>
<tr>
<td>WSO2 Identity Server</td>
<td>v3.0</td>
<td>Core, Multiple Decision, JSON, REST</td>
<td>Java</td>
<td>Apache 2.0</td>
<td>✓</td>
</tr>
<tr>
<td>FIWARE AuthzForce CE</td>
<td>v3.0</td>
<td>Core, Hierarchical RBAC, Multiple Decision, JSON, REST, Data Loss Prevention / Network Access Control, Addition Combining Algorithms</td>
<td>Java</td>
<td>GPL</td>
<td>✓</td>
</tr>
<tr>
<td>OpenAZ</td>
<td>v3.0</td>
<td>Core, Multiple Decision, JSON, REST</td>
<td>Java</td>
<td>Apache 2.0</td>
<td>✓</td>
</tr>
<tr>
<td>Axiomatics Policy Server</td>
<td>v3.0</td>
<td>Core, Multiple Decision Profile, JSON, REST, Hierarchical RBAC, Hierarchical Resource, Privacy Policy, SAML 2.0, XML Digital Signature</td>
<td>Java, .NET</td>
<td>Commercial</td>
<td>✓</td>
</tr>
</tbody>
</table>

**TABLE 1**

ANALYSIS OF IMPLEMENTATIONS


