Federated Authorization Best Practices

- Recommended Approach for "In-Band" Authorization
  - Handling Groups and Non-Unique Values
  - Delegating Entitlement Identification
  - A Note Regarding eduPersonAffiliation

- Examples
  - General Library Access
  - WebAssign

- Appendix A.
  - Background
  - Definitions
  - Challenges and Incentives
  - Solution Patterns
  - Considerations for "In-Band" Authorization

This is a DRAFT for consideration.

Recommended Approach for "In-Band" Authorization

The `eduPersonEntitlement` attribute or "claim" is the recommended way to carry authorization data along with ("in-band" to) authentication. It is also suitable for use with supplemental lookup approaches such as LDAP, SAML Attribute Queries, or OpenID offline scopes, to name a few.

This attribute does not itself define any specific values for authorization, but is defined to carry only URIs so that its values are inherently unique and unambiguous. It supports (but does not require) the use of a registry of shared values, so it scales to address both shared and service-specific values.

When considering a new use case, deployers should review any registries of common entitlement values for any that may match, but do not bend or force-fit a definition if it doesn't suit. When creating new "shared" values, one should generally do so in as small a scope as practical initially and expand the scope as it becomes beneficial to do so.

Note that REFEDS maintains a registry for all of `eduPerson` that includes standard `eduPersonEntitlement` values.

Both URLs and URNs have advantages and disadvantages as entitlement values. URNs tend to work well when creating "shared" values that expand beyond the scope of individual services. URLs tend to work well for service-specific values.

Handling Groups and Non-Unique Values

Groups (and roles, used interchangeable in this discussion) are a common means of associating application permissions with sets of subjects, but they generally are named in ad hoc fashion. Frequently when LDAP is used, groups may be collected under application-specific OU "trees", or at least identified via Distinguished Name (DN), which makes them potentially unique but not readily mappable into a URI or suitable for use outside the administrative domain or in modern federated protocols.

In addition, the "short" names of such groups frequently collide with each other. Consider how often group names such as "admin", "user", or many other similar names tend to be encountered when managing systems. Now consider what happens if such a group name is supplied to an application and an accident of configuration causes membership in the "admin" group for one service to be supplied to another. This is why URIs are a necessary precaution.

At least for service-specific groups, it is a suggested practice to use a service's unique identifier as a URL prefix for service-specific entitlement values. Turning "raw" group or role names into entitlements by prefixing them in this way makes it very easy to create automated rules for both constructing the values and for limiting data release. This approach works well with SAML when sensible entityIDs are used. It also works with OpenID Connect provided an appropriate client ID is used, or alternatively a "scope" could be created in the form of a URL to both identify the right data and prefix the values.

Scopes are typically not URLs but certainly can be.

For example, if a service is identified as "https://sp.example.org/saml2", then entitlements might be constructed by an Identity Provider with values like:

- `https://sp.example.org/saml2/admin`
- `https://sp.example.org/saml2-publishers`
- `https://sp.example.org/saml2/viewers`
- etc.

Note that this does not constrain an application's own naming scheme for groups or authorizations. If required by application constraints on group names, mapping these values back into locally accepted ones is straightforward. Suffixes can certainly be chosen to map directly to the local names if this information is available. On the other hand, simply supporting such group or role names directly is advisable and should cause no particular difficulty.

Delegating Entitlement Identification

While the advice in the previous section is appropriate for service-specific scenarios, it does not account for entitlements with a broader scope and may still require some degree of coordination. It tends to work best (though certainly not exclusively) within the enterprise or for software-as-a-service deployments.
When lacking a tight coupling between the management of a service and the administrative domain(s) providing the authorization values, services should consider providing user interface features that allow each administrative domain (i.e., each Identity Provider) to provide as a configuration setting the entitlement value(s) to map into particular local groups or permission sets. This allows the establishment of entitlement values to be completely delegated to the administrative domain asserting them, getting the service out of the business of worrying about the particulars. This approach requires more up front development, but provides the least ongoing cost for everyone.

This approach is of course entirely compatible with the prefixing suggestion above.

Alternatively, it is generally better for services to establish the values to use than to leave this up to each administrative domain because that will quickly become unmanageable for the service operator. This also affords the opportunity to identify appropriately-defined shared values if possible.

A Note Regarding eduPersonAffiliation

While it is a common practice historically to use eduPersonAffiliation (or eduPersonScopedAffiliation) for authorization, the values of this attribute are not precisely defined by design, and vary quite a bit across organizations and cultural contexts (e.g., the meanings of "staff" and "employee" can be very surprising due to language differences). While there are scenarios in which it may be sufficient to approximate an authorization rule with some of the more commonly understood affiliation values (particularly "member" and "student"), this should never be done with resources of any significant value because there will be substantial numbers of exceptions on both sides of the line with any rule based on them.

Using an entitlement instead allows the home organization to use their internal affiliation data to populate a value for the majority of cases while identifying exceptions that should (or shouldn't) get the value at the same time, providing a much more accurate answer. Unfortunately, supporting both options at once is difficult unless a rule can be applied that recognizes which organizations wish to rely on entitlements; the absence of an entitlement value may cause a service to fail back to the wrong answer derived from an affiliation. This would, though, at least allow for positive exceptions to be handled easily, if not negative ones.

Examples

General Library Access

The most commonly federated authorization use case is library resource access under "standard" contract terms that cover most of an institution's active community and those physically present at a library, but typically excludes guests and some other types of non-traditional affiliates. An eduPersonEntitlement value of "urn:mace:dir:entitlement:common-lib-terms" was defined in 2003 to address this use case and should be used any time this kind of arrangement applies.

Organizations can apply this value to the appropriate people without regard for the particular service being accessed since it is by design a general value that can be applied to any service that uses this kind of standard contract language.

Note that this value is not meant as a generic signal to any library (or other) service that "access should be granted". If that were the meaning, then every single use of the value would have to be individually managed to ensure it aligns with the terms of use for a given service. It is not intended on its own to signal that a given organization's users have access to any given service. As with the (mis)use of affiliation, considerations of organizational access, billing, etc. have to be addressed separately.

The point of a common definition is for the home organization to be able to apply it on the basis of who the subject is, not what the service is. Misusing this value to apply to contractual arrangements other than intended undermines the purpose of the common value, just as misusing affiliations for authorization undermines the ability of organizations to freely release those values without examining every service's (mis)use of them beforehand.

WebAssign

WebAssign is a Learning Management System, and was an early adopter of SAML. Like most applications of this sort, it requires the ability to group students and instructors into courses or sub-groupings of courses. They also chose to allow for in-band enrollment of students. To do so, WebAssign followed one of the patterns described above and provides a text box for each grouping of students that course administrators can fill in with an eduPersonEntitlement value to look for to populate a student into the course. This has worked extremely well for a number of universities that generate automated entitlement values based on student enrollment data and requires no additional work on the part of WebAssign to accommodate any university's particular approach to producing those values.

Appendix A.

The remaining material provides additional background, terminology, and discusses the various solution patterns commonly seen for this problem. It is helpful in understanding the recommendations, and things to consider with other approaches.

Background

Most of the work undertaken over the last two decades or more in the area of federation has been focused almost exclusively on the problem of authentication, identifying subjects and data associated with them, largely data that exists independently of a subject's relationship with a particular service. Considerably less time and attention has been expended on the authorization problem. This is partly because authorization is much harder than authentication. Many services do very rudimentary authorization, if they do it at all, and by far the most common approach to authorization involves maintaining raw lists of users in application-specific databases for use by only a single application at a time. Sharing rules for authorization across applications has never been terribly widespread or successful even within the enterprise (as anybody involved in an effort to define "roles" can attest).

Adding federation introduces a layer of business complication that often defies solutions. Few services have the practical ability to delegate authorization to other administrative domains because those domains have no knowledge of, or interest in, the authorization problems of applications for which they are not responsible. Even if a few would be willing, the need to get most or all of the home organizations of an application's users to manage their access to such applications has proven intractable, and the difficulties tend to heavily depress any interest in this problem.
Nevertheless, there are situations in which the only practical way to manage access to a federated service is by the organization responsible for authentication. While this is most common chiefly in "software-as-a-service" scenarios which do not truly meet the definition of "federated", there are cases in which the sheer scale of use requires that authorization be managed externally by the users' home organizations, and some of these cases are federated in the truest sense. Thus, even if the relative number of authorization use cases remains somewhat small, it remains important to establish good practice around how to solve this problem in the most standardized fashion possible.

Definitions

A few definitions are useful in gaining a clear understanding of why this problem can be so difficult.

User/Subject – These terms are used interchangeably to represent the entity accessing an application or service. Some subjects are "users" in the sense of being people but the concepts addressed by this document apply equally whether a subject is a person or a service account. Subject is the generic term of art.

Administrative Domain – An organizational boundary within which there is direct control or coordination over users and management of identity and access.

Attribute – Overloaded term (used particularly with SAML) for how to communicate a discrete piece of data about a subject. Other protocols (and SAML vendors) use the term "Claim" to mean essentially the same thing. Corresponds in many cases to the attributes on an LDAP entry in a directory.

Identity Provider – The source of authentication and authorization information in a federated protocol.

Service Provider – The target of authentication and authorization information in a federated protocol. Often termed a "Relying Party", or (in OpenID, confusingly) a "client".

Federated Authorization – A deployment in which control over who can access an application/service is managed by an administrative domain different from the one that controls or operates the service.

Challenges and Incentives

There are "degrees" of federation that impact the scope of the authorization problem. When an application is operated by one organization on behalf of another, and the data belongs to the organization managing the users, this is federated in practice but not in spirit. This is because the incentives for managing access properly lie with the user-managing organization rather than with the application-managing one. It also tends to involve only one, or a very small number of, administrative domains of users, which limits the scale of the problem. Getting one or two or three organizations to agree on an approach to something is much different in scale than needing 100 or 1000 or more to agree to something.

A truly federated authorization problem exists when an organization operating a service owns the data or resources and thus has the incentive to properly manage access, but the information and incentives are misaligned. The service owner may have a grasp of the criteria for access control, while the home organizations of the users may have a grasp over which users actually fit the criteria. But if those organizations can't be made to care enough about the service to do work on behalf of the service owner, the problem quickly becomes insoluble at a business level even before the technical challenge is considered. This is why most practical use cases for federated authorization involve contractual scenarios in which managing access is simply a legal/contractual requirement and thus has to be solved.

Solution Patterns

The solutions to the federated authorization problem tend to align around whether the information is delivered in batch or in real-time. In digging into these solutions, one may notice that managing authorization tends to be deeply intertwined more generally with the larger topic of "provisioning". Many solutions for provisioning accounts and keeping them up to date tend to also address (or need to address) the authorization problem as a subset of the larger one. In turn, changing how authorization is done will often require re-examining how provisioning is handled as well.

With a batch approach, by far the most historically-common way this is handled, a feed of data about the users and their appropriate levels of access to a service is delivered on some periodic basis by each organization that manages the user population in the feed. For many years this was the only method commercial services provided for managing access to their services, and is still very common in large enterprises and large commercial services in the HR and Finance sectors. It's the "mainframe" way of doing this: bulky and reliable. The main problems with this approach are the freshness of the data (delaying access or removal of access by hours or days) and the sheer scale of managing feeds for the number of services organizations have outsourced these days. Furthermore, this approach works poorly with truly federated services because most organizations are not likely to be willing to manage and support feeds for services they do not contract for. For most service operators, though, it is likely the simplest way for them to deal with the problem, leading to resistance to the adoption of more complex approaches.

Real-time approaches to authorization include a number of different possible "channels" to communicate the information. Principally the distinction is between in-band and out-of-band methods relative to the authentication channel. Out-of-band approaches rely on direct communication from site to site to create, update, and delete information in a target system. These approaches can be complex and expensive to maintain (arguably moreso than batch feeds), and most importantly they add to the integration cost of a system because they don't address authentication at the same time. Their chief advantage is that they provide a real time view of the "state" of an integration from the perspective of an application. Whether somebody exists and has particular access can be generally known by anyone with access to the application, independently of whether that somebody has ever even accessed the application.

In contrast, the least complex solution with the lowest aggregate cost for all parties, and the subject of the rest of this document, is to communicate authorization information at the same time as, and together with, the authentication exchange when users attempt to access a system. Virtually all modern authentication protocols have the capability to pass additional information about users, including authorization information that can be as up to date as the organization's identity and access management make possible. It then becomes a requirement of the implementation of that protocol on the service side to support the use of that information when users login to an application, often including storing/updating the information in the application for auditing and efficiency purposes.
The key weakness of this approach is the lack of real-time knowledge by the service of the "state" of access management at any single point in space or time. It also provides no means of deprovisioning a system because naturally if a user is gone or even just loses access, they will generally not login to the application to make it known that they in fact don't have the right to do so. This is a particular problem when services insist on charging for every record rather than metering based on "active" access by users. Naturally, there is a profit motive to doing that as well.

Considerations for "In-Band" Authorization

Provided that one accepts the value of an "in-band" (with respect to authentication) approach to authorization, these are some core considerations to bear in mind.

Commonality

The name and syntax of the Attribute used to carry the information doesn't strictly have to be standardized, but as with all uses of federation protocols, doing so helps both Identity and Service Providers limit the need for custom configuration.

Uniqueness

Authorization values either have to be unambiguous in and of themselves to avoid conflicts across services, or every Identity and Service Provider has to be careful to handle the data such that a value meant only for one service isn't accidentally used with another. For example, associating a group named "admin" with a subject isn't terribly safe if one isn't careful to denote somehow which service the group is about. This historically works very badly (because it introduces processing requirements above and beyond the simple handling of data). Using inherently unambiguous values that can't accidentally be misinterpreted is much simpler. URIs (that is, either URLs or less commonly URNs) work very well for this purpose, which is an approach very common to SAML but much less so with other protocols.

Roles vs. Groups vs. Entitlements

A good way to get hung up over how to deal with authorization is to approach the subject from an "academic" perspective that tries to split hairs amongst various ontologies for representing a subject's access to an application. At some very deep level, they're all fundamentally different. At the level where most people operate and try to deploy services, the distinctions seem quite meaningless in practice.

Scope

The scope of an authorization value should be as broad as possible, but no broader. That is, when multiple services (perhaps many of them) can usefully share a value, do so; if not, a service-specific value is appropriate.

Accuracy

Resist the temptation to re-use an existing value that isn't accurate simply because it exists or is deployed for some other service. This is not a good way to approach a use case that matters to either party. If there is honestly so little business value in arriving at the right answer, it is very likely that federated authorization itself is simply not a good fit for the application in the first place.

Privacy

Along with more prosaic considerations such as size limits, services should only receive authorization values applicable to them and not the full set of possible authorizations a subject may possess. This is an important privacy control to prevent information leakage about a subject. This in turn means that it is important to efficiently associate authorization values with services, particularly when dealing with service-specific values that may change frequently. It bears noting that this is impossible to do with some proxy-based approaches to federated authentication because it may be difficult or impossible to know what service(s) a subject is actually trying to access.