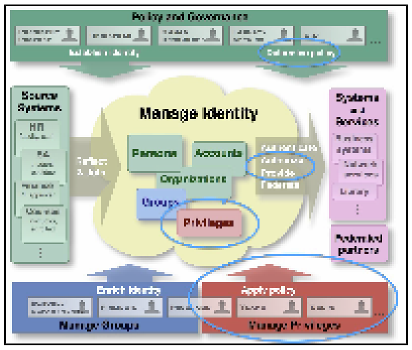
Access Management Recipe V2



Introduction

Identity and Access Management are the two major areas of what is often referred to as core middleware by the Internet2 community. Access Management is the set of policy-based and technology based practices for controlling the access to resources. Access management is in the early stages of its maturity. Its evolution will likely follow the path followed by the more mature area of identity management. "Single sign-on" (SSO) began to take shape in the 1980s with MIT's Project Athena and Carnegie Mellon's Andrew Project. Before then it was standard practice for each application to manage its own set of username and password. In the 1990s Sun Microsystems introduced the Network Information Service which was that vendor's attempt to provide a single sign-on across a common managed set of UNIX systems. Eventually the number of services and applications grew and SSO became a requirement in most enterprises. Today Kerberos, Active Directory, CAS and others fill this technology space to provide "single sign-on" in one enterprise. Shibboleth is now enabling SSO to cross enterprise boundaries and enable virtual organizations.

The current state of Access Management parallels the early days of single sign-on. The granting and management of access to resources is still mostly managed by individual application owners. The current movement toward centralized access management is being led by technologies like directories and Grouper. Identity Management gained maturity not only because of various authentication technologies but because of the common management practices around identity attributes. Authorization attributes, those used primarily to grant access to resources, will need to be managed in a similar way. Common access management protocols will have to be agreed upon and *standard access management attribute profiles* will have to be determined. Given our profession's experience and increasing security and privacy concerns, we doubt this will take the 20-30 years Identity Management did to reach maturity. The stages of access management, looking in both directions before crossing, look like this:

1. Authentication only, if you can get access you get everything

2. A user agreement saying you won't abuse the information you see (everyone with root gets this)

3. Hard coded subject to privilege tables within the application

4. Hard coded subject to privilege tables within the application with some LDAP calls embedded

5. Subject to privilege tables within the application and provisioning of privilege tables from a Grouper-like service with some LDAP calls embedded

6. A centralized service for obtaining all interesting attributes beyond those typical found subject oriented directory - privileges or group names denoting privileges

7. A centralized XACML-like service which essentially grants or denies access

On Roles and Privileges

The definition of role is a common debate in access management. There are good definitions for role, but many identity management and access management technologies have used the term for their own purposes, thus adding to the confusion. In the Apache Shiro project, a role is a collection of privileges. However, there are two differentiators between that definition and the definition used in this document. A collection of privileges is not necessarily a role. Privileges can have inheritance, so a privilege that implies other privileges might not be a role. Roles generally describe the subjects' affiliation, job function, or responsibility. Roles can inherit privileges from other roles.

The terms permission, privilege, authorizations and entitlements are sometimes used interchangeably. We will use the term privilege to encompass these terms.

A Phased Approach to Getting Started

Access management is a *process* as well as a set of technologies. The Grouper project's experience suggests:

- Start out using a single user attribute (e.g. affiliation) in a campus enterprise directory, and let applications make authorization decisions. See the eduPerson definitions at http://middleware.internet2.edu/eduperson/
- Enrich centralized access management using groups determined from "systems of record" such as course memberships, those that can charge against financial accounts, department membership
- Empower departments and application owners to create and maintain groups for their own purposes and unburden central IT of this responsibility
- Support other deeper integrations with access management, support access decisions via web services, provide access to predefined roles and privileges. Examine opportunities for provisioning privileges and group assignments into applications that cannot reach out to supporting infrastructure

Further we recommend:

- Examine your environment for high-value access management use cases that cannot be solved simply by using groups or roles and relatively static attributes
- Look for opportunities to capture policy decisions around access management into access management infrastructure rather than individual applications

Definitions

The **Action** is the verb of the privilege assignment which allows a resource to be assigned to a subject in various ways without creating more resources . For example SubjectA can view (action) the Math department data (resource).

A **Claim** is a kind of attribute used for authorization purposes that is asserted by an outside process.

A **Group** is a collection of subjects. An example of using a group without using authorization is an email list. This being said , a group is often used to represent a privilege set or role with the name of the group or its position in a hierarchy denoting the privileges belong to the collection of subjects. In its purest sense a group is just a set of like objects but historically it has been used for many implied purposes.

A **Limit** is a condition on the privilege assignment which must be true at run-time for the privilege assignment to be allow. Examples of limits are time of day, source IP address, amounts of approvals, etc.

A **Privilege** is an expression of access to a resource.

A **Privilege Assignment** associates the subject with the actions and resources that they are allowed to perform.

A **Privilege Set** is a collection of privileges (see Privilege Assignment) that is shared by all subjects or roles assigned to the set

A **Resource** is the part of the system which needs to be protected by authorization, and it represents a noun in a privilege assignment.

A **Role** is an object assigned to subjects which describes the subjects' affiliation, job function, or responsibility. Privileges associated with the role are effectively assigned to the role's subjects. Roles can inherit privileges from other roles.

A **Rule** is computation done on attributes by either a resource or a Policy decision point to grant or deny access to a resource.

A **Subject** is a person, a software component (service) acting on behalf of a person or a set of subjects and/or services.

Inheritance Definitions

Many of the parts of the privilege model can have **inheritance** from other objects. The privilege management system might require the inheritance to be a hierarchy, or maybe it is required to be acyclic, or maybe it could be a directed graph.

- **Implied Group Membership**: If the system allows a group to be added as a member of another group, then membership in a group implies membership to other groups.

- Privilege Set Inheritance means that privileges assigned to a subjects in a privilege set are inherited to other privilege sets. For instance, to reuse the definition of such a set, a set "SeniorLoanAdministrator" could be allowed to do everything that a subject assigned to the privilege set "LoanAdministrator" can do, plus a few more privilege assignments. Note that if there is privilege set inheritance, it does not mean that a subject assigned to "SeniorLoanAdministrator" is also assigned to "LoanAdministrator".

- Action Inheritance means that assigning an action to a subject for a resource also implies that subject has other actions on that resource. For instance, subjects with Action<Write> on a resource might also implicitly be able to Action<Read> the resource. In other cases, the read action might need to be assigned separately from write.

- Resource Inheritance means that subjects which are assigned an action on a resource have other implied resources for that action (and inherited actions). An institution's organization chart or course structure could be modeled as privilege resources. Instructors could be granted privileges on individual courses, and deans could be granted privilege actions on entire departments or schools. (Sometime a resource which implies other resources has been referred to as a role, however, it might be that the bundle of resources qualifies a role instead of defining new role.)

Access Control Decisions (groups, roles, privileges, and external authorization)

Overview

Applications have various access control needs to control which subjects have access to what. This is generally for applications which require authentication, though the authorization could also apply to anonymous subjects (who would get privileges from an anonymous role).

There are several decisions for access control for applications including:

- groups vs roles
- roles vs privileges vs hybrid
- hard-coded privileges vs externalized authorization
- for externalized authorization: centralized or not
- caching

Groups vs Roles

When applications protect resources by checking if the authenticated user is in a group, they are essentially using a group as if it were a role. For example, if the application code checks if the authenticated user is in the institution's "student" group, in order for them to see the main screen of the application, then there is an implicit hard-coded privilege resource of "main-screen" and an action "view", assigned to the role "studentUser", which is assigned to the group "student". Though it is referred to as security by group, it is actually a role.

Roles vs Privileges vs Hybrid

Generally, an application will have a handful to a couple dozen roles. These application roles are used as a security template that multiple subjects can be assigned to for common access. These roles are generally mapped to job function or affiliation. For example, there could be roles for student, staff, instructor, admin, billing administrator, anonymous user, etc. Note that the set of roles varies across applications as needed. Within applications, roles tend to be named and managed internally. Yet these application-specific roles are often derived from larger definitions that have an enterprise scope. "Student" defined via the eduPerson attribute with its global meaning, is likely to lead to the assignment of application-specific "derived roles" which may or may not contain the word student.

If not all subjects in a role have the exact same access, then they need **personalized privileges** assigned to them. The authorization system might assign privileges directly to the subject, or in the context of an application, or in the context of a role. However this is done, the subject will have privileges that are different from other subjects assigned to the same role. An example is to specify which data-sets the subject has access to when searching for data or running reports.

If the authorization system supports **applications with subjects that select one role at a time** to use the application, then individual privileges should be able to be assigned in the context of one role. The subject will not need to have elevated privileges at all times, and thus will be less likely to accidentally perform tasks that only the elevated role can perform.

Hard-coded Privileges

If there is no access control framework or central system, then **hard-coding** the privileges in the application might be the easiest way to assign access. This would require the application to maintain its own privilege-to-subject map. This is actually the current "state of the art" in most applications. Some are able to read ldap directories for attributes or group memberships. You often see the terms "coarse-grained" and "fine-grained" authorization used to describe this kind of situation.

External Authorization

An **external privilege management system** separates the privilege resources from the code, but still runs in the same application for that application.

A **central authorization system** maintains the roles and privileges in a central location as middleware for multiple applications.

External or centralized authorization gives the application run-time flexibility for changes in access control policy and authorization reporting.

Advantages of Centralized Authorization

Centralized authorization systems can show a subject's privileges across applications and can ease revocation. Certainly as auditing requirements increase, and sharing of access control policies across multiple applications increases, centralized authorization is required. If there is a common on-boarding workflow application, then centralized authorization can make the architecture more homogeneous.

Caching Considerations

When using externalized privileges, there are **caching considerations**. Caching can improve the performance of the application and reduce the dependencies of middleware components. If there are not real-time updates from the authorization system, then the privileges can become stale. When there are a lot of privilege resources and assignments which need to be checked, it is a good idea to cache the application's privileges for the entire user population, or for one user when they authenticate. If there are reports or queries which need to join available data with the allowed record types, the authorization information might need to be cached directly in the application's database. If there are limits on the authorizations, then it is more complicated than just a list of allowed action/resource pairs.

Naming Considerations

The grouper project has collected some best practices in naming groups and privileges. These practices can be found at Group and folder design ideas

Basis for Authorization Decisions

Access management decisions can be made on attributes. The approaches below are simply categories of attributes. Normally we differentiate between attributes that are relatively static and those that must be determined at run time. Run time attributes are usually described as limits.

- Attribute based Approach (What you are)

- Claims based Approach (What someone says about you)

- Group/Role based Approach (What you belong to)

- Rule based Approach (What the application computes about you)

For more details see

https://spaces.at.internet2.edu/display/macepaccman/Authorization+Techniques+and+Strategies

Provisioning Access

The process of transporting attributes, privileges, groups, roles, and subjects etc. to a resource that does not participate in the central identity and access management solution. For applications that rely on hard-coded privileges as described above, provisioning these privileges can be an acceptable compromise to make the application more dynamic.

Getting Your Attributes

At a conceptual level there are many places where attributes can be obtained

- From a directory that provides attributes from well understood schemas via LDAP
- Derived from a SCIM or a SAML attribute assertion
- From an XACML Policy Information Point or as a result of a call to a XACML authorization API

Different programming and application environments may hide these details from a resource programmer much like the CGI variables are presented to a web programmer even though they are passed via HTTP. Most application environments have the ability to utilize LDAP apis or make web service calls via SOAP or REST

Are there different attribute types?

Some attributes are part of a subject's identity and we refer to as ***intrinsic attributes***. These attributes can be used to identify a subject and differentiate subjects from one another. These attributes are normally mastered at a system of record, usually an HR system or a student system.   ***Applied attributes*** are generally based on "claims" as mentioned about and are attributes asserted about a subject. These attributes are generally derived from some sort of workflow where a sequence where people make claims within a auditable process . The distinction can be subtle. As for example eduPersonAffiliation is intrinsic and is determined by a system of record and is a byproduct of an identity proofing process. The attributes isMemberOf and eduPersonEntitlement are more clearly applied attributes . Another kind of attribute might be an ***environmental attribute*** or more properly ***limits*** , these attributes tend to not be related to the subject but are attributes of the surrounding environment and are usually calculated in realtime. Attributes like time, temperature, "class roster full" fit into this category.

A XACML Perspective on Access Management

From Wikipedia:

> *"XACML stands for* eXtensible Access Control Markup Language. The standard defines a declarative *a*cess control policy language implemented in XML and a processing model describing how to evaluate authorization requests according to the rules defined in policies."

XACML is an OASIS standard and uses the following terminology

Terminology	**ComparableShibboleth Component**	**Comparable Grouper Component**

Policy Administration Point (PAP) - The location which administrates the policies	Available on both the IdP and SP configurations in XML files	The policies can be administered in the Grouper UI, web service, loader configuration, etc
Policy Decision Point (PDP) - The location which evaluates and issues authorization decisions	Shibboleth IdP as the IdP may forward attributes to the SP which are used in enforcement	The Grouper web service can evaluate if someone is a member of a group /role, or has certain privileges, etc. It can also take into account limits if applicable when computing the result.
Policy Enforcement Points (PEP) - The location which intercepts the user's access request to a resource and enforces the PDP decision	Shibboleth SP module	You can use the Shibboleth SP module, or code in your application could make an LDAP call, or a web service call to Grouper to check privileges (the GrouperClient could help via Java or command line)
Policy Information Points (PIP) - The location which provides information the PDP	LDAP directory or SQL database connected to the IdP	Grouper can pull data from external SQL datebase or LDAP data sources via the Grouper loader and act a PIP for its clients via the Grouper web service or though a provisioning method.

The Grouper Perspective on Access Management

http://www.internet2.edu/grouper/

Integration with Cloud Resources

Google Docs example The Google Apps API reference Guide

Using scim profile as a source of shared attributes

Provisioning ERP solutions

Provisioning Peoplesoft at Madison

The NC State Peoplesoft use case.

Penn Groups

Kuali Identity Management

Workday

Standard and not quite standard APIs for Authorization

The grouper APIs

XACML APIs

SAML attribute request API

Opensource Identity Management for Higher Education

Packaged solutions

Oracle Entitlement Server

Grouper

perMIT

spocp

permis

Axiomatics Policy Engine

===================

Link to older version of the recipe.

Link to LDAP Recipe

MACE-paccman User Case Library