Authorization

Overview

Fedora 4 authorization is designed to be fine grained, while at the same time manageable by administrators and end users. Authentication is managed by the servlet container or externally, but authorization mechanisms are open to extension and many reference implementations are included. Roles-based access control is an included feature that makes permissions more manageable and at the same time easier for external applications to retrieve, index and enforce. Finer grained security checks have no impact on the performance of requests that have a Fedora administrator role.

Use Cases

1. Researchers control the polices on their own objects
2. Distributed authentication and authorization
3. University of North Carolina at Chapel Hill
    a. Unified Authorization
    b. Setting Individual Permissions
4. Yale University
    a. Fedora managing access conditions
    b. Programmers use API for access condition support in external systems, i.e. HydraTitle (goal)
    c. Applications use API for updating access conditions stored in Fedora
5. University of Wisconsin - Madison
    a. External authentication and authorization
6. Islandora
7. Hydra
8. Avalon Media System

Container Authentication

User authentication is generally handled by the Servlet container, i.e. Tomcat, JBoss AS, Jetty, etc. Authenticated requests will arrive at Fedora servlets with a non-null values for getRemoteUser() and getUserPrincipal().

Fedora users may have the following servlet container roles.

- **fedoraAdmin** - Grants superuser permissions to the the Fedora repository. Bypasses the configured policy enforcement point (PEP).
- **fedoraUser** - The policy enforcement point (PEP) grants permissions on the basis of this authenticated user and the credentials on the request.
- **fedoraProxy** - The policy enforcement point (PEP) grants permissions on the basis of end-user security credentials that are forwarded via extended request headers.

Extension Point: Container Authentication

Implementations may configure the servlet container to employ any user authentication mechanism that meets specifications. This is container-specific, but usually includes JAAS, LDAP, CAS, Shibboleth, etc..

See the overview on configuring servlet container authentication for more details.

Additional Security Principals

Access may hinge on additional security principals that are specific to an organization. These principals are often based on Shibboleth, LDAP, CAS, databases and other sources. Additional principals can be included in authorization by implementing a PrincipalFactory. A PrincipalFactory examines Servlet requests and returns a set of additional principals. Examples include a named IP range, an affiliation or group from a Shibboleth header, principals extracted from SAML payloads, etc.. Fedora provides a configurable HeaderPrincipalFactory that extracts principals from headers.

Extension Point: Principal Factory

Implementations may enhance the security context for all authorization decisions downstream by implementing a Principal Factory, which extracts additional security principals from servlet requests. Principals are extensible to whatever credential the organization wishes to privilege. Principal names must be unique.
Reference Implementation: IP Range Principal Factory

In Development: Fedora ships with a principal factory for named IP ranges. The factory may be configured with a map of names to a set of IP ranges. This allows Fedora administrators to assign privileges to all users within a named IP range, such as "On Campus".

Reference Implementation: Header Principal Factory

In Development: Fedora ships with this simple principal factory that creates string-based security principals from request headers. This is useful in cases, like the Apache HTTP Shibboleth module, where additional attributes are supplied as request headers.

Forwarding Security Credentials

Since applications often act on behalf of end users with extended security attributes, such as those from Shibboleth, the ability to forward credentials to a central point of authorization is key. Regardless of the approach used for *authentication* of third-party applications, these applications will need to forward security attributes on behalf of end users for *authorization*. An example of this pattern is the X-Forwarded-For header often used in web proxies, which we can use to forward the end-user IP address.

Fedora supports both end users and application users. It is helpful to feed both local and forwarded security credentials into a same pipeline for extracting security principals that are the basis for authorization.

Access Roles API

The access roles API allows you manage the assignment of access roles throughout the repository tree. For details, please see the Access Roles Module.

Fedora Policy Enforcement Point (PEP)

Fedora includes an extension point that allows installers to build their own enforcement logic for all Fedora actions. A PEP enforces appropriate access for fedora users and their proxies, i.e. applications acting on their behalf. The PEP interface is simple, for details please see Authorization Delegates. Some policy enforcement points may be roles-aware, meaning that they leverage role assignments from the Access Roles API.

Extension Point: Policy Enforcement Point (PEP)

A policy enforcement point enforces appropriate access for Fedora users and their proxies, i.e. applications acting on their behalf. One policy enforcement point may be configured at a time.
Reference Implementation: Basic Roles PEP

The basic roles enforcement point determines access on the basis of 4 simple roles that may be assigned throughout the repository. These are reader, metadata reader, writer, and admin. For details please see the Basic Role-based Authorization Delegate.
Reference Implementation: XACML PEP

In Development: The XACML PEP forwards authorization requests to a XACML policy decision point. It is aware of access roles and may also make determinations on the basis of a wide range of Fedora resource properties. Policy sets may be customized for different part of the repository tree. For detail please see the XACML Authorization Delegate.