Purpose

The purpose of this document is to outline the implementation of the IP/NetID user cases defined in the IP/NetID User Case Specification [reference]. This includes restricting access to an entire object and/or certain datastreams based on either a user's authenticated credentials or IP address.

What are the types of restrictions?

An object might need to be restricted from discovery through the normal search and discovery mechanisms that are used, the RUcore search portal application and corresponding APIs. This restriction can be based on client IP address or the authenticated credentials of a user.

Access to one or many datastreams of an object might also need to be restricted based on the users' current authorization credentials. The resource might still be discoverable; however access to the datastreams would not.

Expanding our thinking not only access to a datastream, but the type of access may also be limited based on that users credentials. Types of limited access might include:

- Downloading of a datastream or set of datastreams
    - Archival masters
    - Presentation video files or audio files
    - Restricted documents in PDF form
- Print permission of a datastream or set of datastreams
    - Restricted documents
- Selecting content of datastreams
    - Selecting and copying text from a document

Embargoes of datastreams exist based on a time parameter. This restriction is accomplished using the native XACML policy features of Fedora. This is another type of restriction.

Restricting access to a resource

Current process

Currently we restrict access to a resource by making it part of a special search portal, a shadowed resource portal. When a resource is added to this shadowed resource portal the resource also needs to be removed from other portals so that resource isn't discoverable. Whenever a resources page is rendered a check is made to determine of the resource is part of the shadowed resource portal. If it is, then a message is displayed stating that the resource is restricted and no resource information is shared.

Proposed Solution

To restrict discovery of an object the first place to start is with the search index. If the resource is indexed in a way that is not discoverable through normal means then it will not be part of a queries result list.

To accomplish this I am proposing a new index field be added to the Solr index of each object. When a user submits a query, that action triggers a check of the user's credentials; both user account and IP information are submitted to Solr as part of the query. If the submitted criteria do not match objects in the Solr index, the object should not be returned as part of the result set. The same mechanism that is used to determine the entries in the new Solr index field can be used when rendering a resources page. If the user current credentials do not match the requirements for the resource, a message stating the resource is restricted can be displayed. This would replace the shadowed search portal capability.

The values in the new Solr index would be abstractions of IP addresses or ranges for a couple of reasons. Indexing and searching for individual IP addresses would be simple, but adding in ranges complexity would increase significantly. Single IP addresses might not change frequently, however ranges might expand or shrink frequently. With any change in an IP address range the values that are stored in the Solr index would need to be updated, requiring a re-indexing. Abstracting the range to a simple value would mean that any changes to IP addresses could be made outside of the Solr index and applied when mapping the user current IP address to the list of possible abstractions that it matches on.

Examples

A resource with no access restrictions uses a public value as a default.

```
<field name="access">group_public</field>
```

A resource's Solr index that has had its access limited to a single IP 198.151.130.130.

```
<field name="access">ip_mills-chad-tsb</field>
```

A resource's Solr index that has had its access limited to IP addresses 198.151.130.*

```
<field name="access">ip_tsb-building</field>
```

A resource's Solr index that has had its access limited to IP addresses 198.151.130.* and

198.181.6.1-198.181.6.64 and 96.234.41.179 and rutgers-faculty user group access.

```
<field name="access">ip_tsb-building</field>

<field name="access">ip_scc-department</field>

<field name="access">ip_mills-chad-home</field>

<field name="access">group_rutgers-faculty</field>
```

When a query is submitted from a client, the client's IP address will be mapped and the abstractions will be submitted with the query.

If a request was made from the 198.151.130.130 IP address, the abstractions that would apply would be group_public, mills-chad-tsb, and tsb-building.

If a request was made from 198.151.130.100, the IP address abstraction that would apply would be tsb-building. For IP address 198.181.6.65, the abstraction would be only group_public; since it doesn't match the IP range criteria for the scc-department of 198.181.6.1-198.181.6.64.

Implementation

To implement these abstractions, every resource will need to be re-indexed. Any new abstractions or further application of those abstractions to resources will require a re-index of the resource. This is very similar to the re-indexing process that takes place when adding and removing resources to and from search portals.

By default, any publically discoverable resource would have an access entry with the value "group_public." Perhaps for any resources part of a dark archive that default access wouldn't occur, making the resource undiscoverable through public interfaces that user the search API.

When submitting a query through the search API, the abstractions from the client's IP address and the user credentials, if authenticated, will be passed in as part of the Solr query.

Restricting access to a datastream

Current process

Currently, datastreams can be embargoed using a Fedora XACML policy. All access prior to the embargo date expiring is limited to the fedora admin user. The user interface is tipped off to any embargoes by the datastream manifest for a resource provided by the get API. The get API interacts with Fedora and when discovering the resource has an XACML policy it tests to determine if the policy is still being enforced. If the embargo is still active, the get API responds with the appropriate http status code and @reason.

```
<file>
      <id>PDF-1</id>
      <label>PDF-1</label>
      <status reason="date">403</status>
      ...
</file>
```

In this example, access to the PDF-1 datastream is currently forbidden (403) for the @reason = date.

Alternatively, a successful response for a datastream that is not embargoed takes this form.

```
      <status>200</status>
```

Proposed Solution

The get API not only provides a datastream manifest for a resource but is also used for all datastream access by public users. When accessing a specific datastream, the user current credentials will need to be tested before access can be granted.

When creating a datastream manifest the user credentials will also need to be tested against each available datastream associated with the resource. To enforce restrictions on datastreams for other reasons (IP address or user credentials), the get API will need to relay the reason for the restriction using the same mechanism. Providing an accurate reason for the restriction will inform the interface what to tell the user regarding the restriction.

An example of an IP restricted datastream would be:

```
<status reason="location">403</status>
```

An example of a user credential restricted datastream would be:

```
<status reason="credential">403</status>
```

Dovetailing the new checks based on user credentials, together with the current XACML policy check using Fedora, will create a more complex and lengthy process for the get API.

Possible Software Solution – Fedora; XACML and FeSL

Using the capabilities native to Fedora, it is possible to write XACML policies that can enforce access based on a client's IP address. There are a number of reasons why using XAMCL is undesirable.

1) We currently restrict access to the Fedora APIs to localhost only. This means that a public client never directly interacts with Fedora. All access is funneled through the get API. When funneling the traffic, the IP address seen by Fedora is that of the localhost, and not the client.
2) Writing IP based restrictions for single IPs in XACML isn't terribly complex; however, IP ranges would be.
3) Policy writing for ranges would need to be handled by a single XACML policy that is related to the resource(s) that use it. Duplicating the information in an individual XACML policy isn't scalable.

For user or user group authorization the Fedora Security Layer(FeSL) would need to communicate with the RUcore SSO application as if the RUcore SSO application were a LDAP authentication point. This is due to the nature of the RUcore SSO accounts not being limited to the Rutgers LDAP users. Once authenticated into FeSL, the RUcore SSO application would need to distribute user credentials to FeSL. Authorization to resources would need to be provided by some complete XACML policy written for the resource, while datastream access would need to be provided using an XACML policy for the datastream.

Having established this use of FeSL for datastream and resource access based on user credentials, the issue still exists that a user never directly communicates with Fedora; there is always a handoff between the user and Fedora using the get API. The Fedora user file would be used to grant the user individual user access or group access to resources or datastreams. Relaying any resource access restriction to the Solr index would need to be added.

Looking ahead with Fedora 4

In Fedora 4.0, FeSL support, and possibly XACML support, will be limited or replaced. Currently the Fedora 4 community has been building some user cases for authentication and authorization support. Some of those user cases look very promising when comparing them with our needs. See the use cases provided by University of North Carolina, the University of Wisconsin, and Yale.

https://wiki.duraspace.org/display/FEDORA40/Authorization

Possible Software Solution – Database solution with API's

A database solution would provide simpler management of access restrictions whether they are based on user credentials or client IP addresses. The solution would need to provide an API that could provide the following tools.

Unauthenticated Users

1) Provide the series of abstractions that are associated with the client current IP address. These abstractions can be used when querying Solr.
2) Provide access cues to a resource based on the client IP address. This function would serve as a replacement to the current search portals shadowed portal which limits access to resources.
3) Provide access cues to datastreams based on the client IP address.
4) Provide access information to the get API for the datastream manifest of a resource.

Authenticated Users

1) Provide access cues to a resource based on a series of abstractions that are associated with the authenticated users' credentials.
2) Provide access cues to datastreams based on the authenticated user's credentials.
3) Provide access information to the get API for the datastream manifest of a resource.

Indexing

1) Provide a method that reports all access restriction abstractions associated with a resource when indexing the resource.

Applying Restrictions

1) Possibly provide a method to the WMS to apply access restrictions based on some criteria.

The interface would need to be created for the management of the restrictions. The following are a list of functional requirements for the interface.

1) Create an access restriction for a resource, or set of resources based on collection or portal. The restriction can be based on IP address(es) and/or user credentials, single user or group.
2) Manage existing restrictions; edit, update, delete.
3) Provide reports of restrictions. An example would be "Generate a list of resources that are restricted to Douglass Library only."
4) Trigger re-indexing of resources in Solr when managing restrictions.

Proposed Solution

Of the two possible solutions the database solution looks to be the most promising. Given the higher amount of complexity of the current Fedora solution the database solution is more straight-forward. Also, given the uncertainty of what Fedora 4 will offer by way of authentication/authorization solutions we might very well save ourselves from re-implementing this project when we migrate to Fedora 4.

A good fit for this solution would be the current search portal tool. The tool already interacts with the search API, and uses a database, web services, re-indexing capabilities, and portal assignment or resources.

Other Considerations

If we implement a database solution for access restrictions we might also want to move the XACML data based embargo capability to the database solution as well. By doing so, we would create a single-source/single-query situation when determining a user level of access to a datastream. This would probably provide us with a performance gain since the traditional Fedora XACML policy testing has been a cumbersome process.