General Account Mapping Advice

Many applications or services make use of local accounts to manage authorisation policy and provide a customised environment. This page gives a general overview of the options available for mapping incoming federated identities to local accounts and the provisioning of those accounts.

Contents

1. Introduction

In the world of Moonshot, the Identity Provider of the user's home organisation authenticates that user and sends a yes/no to the Relying Party Proxy of the service hey are trying to access. Thus, there are three points at which account mapping can be influenced:

1. IdP mapping - Information sent by the Identity Provider can be *directly* used by the RP Proxy or service to map to an account on the service;
2. RP Proxy mapping - The RP proxy can take information given by the Identity Provider, apply some transformational logic, and pass this transformed information to the service for it to use to map directly to an account on that service; *or*
3. Service mapping - The service can take the information given to it by the Identity Provider and/or RP Proxy and apply its own transformational logic to map directly to an account on that service.

On top of this, there are a few options around management of the accounts that users get mapped to:

1. Accounts could be created by the service administrator (or some automated process) before the user tries to access the service for the first time; *o*
2. Accounts could be created on the fly the first time the user tries to access the service (Just In Time provisioning).

To illustrate three possibilities, consider the following example:

A user from the University of Camford is attempting to access an SSH server at the University of Oxham. Their Camford username (NAI) is james.bond@camford.ac.uk, and they have access to an SSH account named "guest007" on Oxham's Moonshot-enabled SSH server.

The three possibilities for doing this mapping corresponding to the options above would be for:

1. IdP mapping - When Camford's IdP authenticates its user, it sends information to Oxham's RP Proxy that the user should be mapped to the "guest007" account. Oxham's RP Proxy receives this and passes that information on to the SSH server, which then logs the user into that account.
2. RP Proxy mapping - When Camford's IdP authenticates its user, it sends information to Oxham's RP Proxy that uniquely identifies the user (either a username or a persistent pseudonymous identifier). Oxham's RP Proxy is configured such that it knows to map the identifier for that incoming user to the "guest007" account, and as such it passes on the "guest007" username to the SSH server, which then logs the user into that account.
3. Service mapping - When Camford's IdP authenticates its user, it sends information to Oxham's RP Proxy that uniquely identifies the user (either a username or a persistent pseudonymous identifier). Oxham's RP Proxy passes on that identifier to the SSH server. The SSH server is configured such that it knows to map the identifier for that incoming user to the "guest007" account, so it logs the user into that account.

The two account management options would involve:

1. The administrator of Oxham's SSH server could create the "guest007" account before the user tries to access it; *or*
2. The first time the user tries to access Oxham's SSH server, either Oxham's RP Proxy or the SSH server would create an account for that user.

Which of these three mapping options and two account management options is the best depends entirely on a few factors, such as:

- Whether the service itself is actually capable of doing such mapping and account creation (standard SSH is not, for example);
- Whether the IdP is willing to manage all of this mapping logic;
- Whether the service is willing to trust the IdP to do this mapping for it or whether it wants to retain control itself;
- Whether the service is willing to allow accounts to be created in an automated fashion or whether it wants to retain manual control over this.

The rest of this page explores each option in more detail.

2. Mapping Options

2.1. IdP Mapping

2.1.1. Overview

The Moonshot Identity Provider of the home organisation of the user is authoritative for two things - providing authentication service for that user (i.e. telling the RP a yes or no about whether the user has correct credentials on that service), and for providing information about the user linked to the credentials for authorisation or personalisation purposes.

In contexts where the RP trusts the IdP enough, it could rely on the IdP to indicate to it which local account that user should be associated with.

Generally, this option is likely to not be suitable for wide scale deployment as the amount of work required for the IdP in maintaining per-user per-service mappings would be high enough to make it undesirable for most Moonshot IdP operators.

2.1.2. Technical detail

To enable this approach, the Moonshot IdP would need to be configured with two things:

- A datastore of some sort to store the per-user, per-service account mappings. This could be something like a database or flatfile that the Moonshot IdP is configured to look at once the user has successfully authenticated.
- FreeRADIUS logic that takes that takes the account name for the remote system and passes it to the RP in the form of a RADIUS or SAML attribute.

2.1.3. Specific Advantages of this approach

- The Moonshot IdP retains fine-grained control of which of their users can access which systems, and how.
- The RP Proxy and service have less to do as management of the mappings is essentially out-sourced to the user's Moonshot IdP.

2.1.4. Specific Disadvantages of this approach

- Managing the mappings of every user for every service they use is likely to represent a high enough workload that the Moonshot IdP operator would simply be unwilling to do this.
- The RP Proxy and service have no direct control over which external users get mapped to which local accounts.
- The user would have to talk to their Moonshot IdP operator to manage access to every service that wished to be deployed this way.

2.2. RP Proxy Mapping

2.2.1. Overview

The Moonshot Relying Party Proxy receives information from the Moonshot IdP about the user attempting to access the service. This information may include persistent pseudonymous identifiers and/or detailed information about the person (e.g. a username, first name, surname, department).

In contexts where the Relying Party / service wishes to manage the mappings between incoming user identifiers and local accounts, and where the service is unable to do so, the RP Proxy is able to take the incoming information from the Identity Provider, do some transformation and/or make some decisions, then indicate to the service which local account the user should be associated with.

2.2.2. Technical detail

The RP Proxy receives information about the authenticated user from the Moonshot IdP in the form of RADIUS or SAML attribute information. When configured correctly (see Section X), the RP Proxy could transform this in one of several ways, including:

- Taking an incoming persistent identifier for that user and mapping it to a local account name for the service according to mappings stored in a database or file (e.g. user 2b30b510-0544-4fbb-a6de-a641fea48b96@camford.ac.uk maps to the "guest007" account).
- Taking an incoming SAML attribute and mapping it to a local account using some logic (e.g. anyone with the "staff" value from the eduPersonAffiliation attribute from Camford's Moonshot IdP should get mapped to the "camfordstaff" account).
- Generating an account on the fly for that user and storing that mapping (e.g. for the new incoming user "jamesbond@camford.ac.uk" create a local "jamesbond" account and log them into it).

Note that in all cases, the RP Proxy would pass a special attribute to the service indicating which local account to use. Which attribute this is and what method is used is application-dependent.

2.2.3. Specific Advantages of this approach

- Authorisation policy is under the control of the organisation that "owns" the service.
- The Identity Provider administrators do not have to maintain per-service information for each of its users.
- The service doesn't have to maintain these mappings.

2.2.4. Specific Disadvantages of this approach

- The RP Proxy administrator would be responsible for maintaining the mappings for all of the services that it is the RP Proxy for (unless they provide delegated access to the service owners to the set of mappings - e.g. by giving them access to the database).

2.3. Service Mapping

2.3.1. Overview

In contexts where the service wishes to manage the mappings between incoming user identifiers and local accounts, and where the service is able to do so, the service itself is able to take the incoming information from its RP Proxy and make a decision about which local account the user should be associated with.

2.3.2. Technical detail

The service receives information about the authenticated user from its RP Proxy, as originally supplied by the user's Moonshot IdP in the form of RADIUS or SAML attribute information and possibly modified/filtered by that RP Proxy.

What the service does with this information to map the user to a local account is very much application specific.

2.3.3. Specific Advantages of this approach

- Control of the mapping of external users to local accounts in under the control of the service itself - the entity which cares the most about getting it right.

2.3.4. Specific Disadvantages of this approach

- Many services are unable to actually do this at present (e.g. OpenSSH has no facility for mapping of users).

3. Provisioning Options

3.1. Pre-existing Accounts

The first option is conceptually the simplest - accounts are created in advance by the service administrator, and the service or RP Proxy needs to store sets of mappings or logic for which external user(s) are mapped to each local account.

To do this, the service or RP Proxy administrator would need to know the persistent identifier that would be presented to it at the time of attempted authentication by the user. This could be done in a number of ways:

- If the user's Moonshot IdP is configured to release a non-opaque identifier to the RP Proxy (e.g. their username or email address), the user could inform the service administrator of the value to expect;
- If the user's Moonshot IdP is configured to release a pseudonymous identifier:
    - targeted specifically to the service, then the user could attempt to login to the service expecting the attempt to fail. The service administrator could then look at the log files produced as a result of the attempt to find out the value to expect, and set up the mapping. Subsequent authentication attempts should then succeed;
    - targeted to the realm community, then a service (e.g. a simple web service) could be set up in that same realm/community that just echoes back the pseudonymous identifier for that realm/community. The user could then inform the service administrator of the value to expect.

3.2. Just In Time Account Creation

The second option would be for a new account to be created every time a new user (i.e. a user for whom the persistent identifier of whatever kind is used as a primary key for identifying an external user) presents themselves to the RP Proxy/service.

- If the service is capable of doing this, then the details for how to do this will be service specific.
- If the service is not capable of doing this, then the RP proxy may be able to do it (depending on the service). For example, where the service in question is SSH, the RP proxy could store a set of mappings of external users to local accounts in a database, and if a new user presented themselves FreeRADIUS could be made to create an account on that system, store the mapping, then pass that username to SSH to log the user in as.