A Design Space Review for General Federation Management Using Keystone

Dr. Craig A. Lee, lee@aero.org
Senior Scientist, The Aerospace Corporation

First Cloud Federation Management Workshop, December 8, 2014
What’s Up with All this Federation Stuff?

• As cloud computing becomes even more widely adopted, users and institutions will want to securely share resources
• Federation Management is being recognized as the critical capability to make this happen
• While people talk about cloud federation, federation really needs to be possible at every level in the system stack
  • Cloud infrastructure services: VM, storage containers, VLANs, etc.
  • Application-level services: databases, specialized computing services, etc.
• This will enable a huge spectrum of collaborative applications
  • Business-to-business, gov-to-gov, disaster response, …
• What does the federation design space look like?
  • What are the deployment and implementation options?
• We present a design space review for cloud and general federation
  • We use OpenStack and Keystone as our experimental vehicle
  • We use the Virtual Organization (VO) concept as the federation management abstraction
  • We speculate about Federation Agents as an implementation concept
• Final Observations
The Fundamental AuthN/Z Challenge in a Distributed Env

Fundamental Requirements for Federation Management:

Client-Side: *Federated Authentication & Service Discovery*

Server-Side: *Federated Credential Validation & Authorization*
Managing Federated AuthN/Z Using a Virtual Organization Management System

This is a conceptual security and collaboration context that is not owned by any one organization. This could also be called a Virtual Project.
Virtual Organization Abstract Concepts

• A VO is a security and collaboration context not exclusively associated with any one physical organization or site
  – Participating partners agree upon structure, rules and processes
  – A VO partner can be a single person, a group or an entire organization
• A VO has members that are assigned roles and/or attributes
  – Membership roles or attributes grant specific capabilities within a given VO as determined by each resource/service provider
• Partners participating in a VO contribute resources, i.e., data and services
  – They retain complete control over their own resources!
  – Access by VO members can be modified or revoked at any time by both the VO administrator and the resource administrator
• A VO Management System (VOMS):
  – Maintains member identity attributes and authorization attributes
  – Enables resource (service) discovery
  – Enables validation of VO member authz credentials on service invocation
Design Review and Implementation Options

- Centralized
  - Trusted, Third-Party VOMS
  - Easier to implement
  - Centralized state makes revocation easier, faster
  - Single point of failure

- Proxies
  - Avoids the requirement to modify services with a VO PEP
  - Cost is higher latency

- Distributed
  - Trusted, Peer-to-Peer VOMS/Keystones
  - Harder to implement
  - Revocation must propagate -- takes longer
  - No single point of failure

- How does the Keystone Object Model Support these options?
  - How can the elements of the Keystone Object Model be used to support centralized or distributed VOMS?
Current Keystone v3 Object Model
with representative object associations (assignments)
Possible Use of Current Object Model for Federation
with representative object associations (assignments)
Alternative Keystone Object Model for VO-Based Federation
VO = Domain, with Domain-specific Roles

Users

VO Domain

Roles

Projects

Services & Endpoints

User_0

Role_0

Proj_0

Svc_0

User_i

Role_j-1

Proj_k-1

Svc_m-1

User_r

Group

User_i-1

Endp

Endp
A Centralized, Third-Party VOMS: KeyVOMS
Secure Discovery and Access across Multiple Sites

• OpenStack Keystone v3 re-purposed as a stand-alone, VO Management Service: KeyVOMS
  – Domain used as a VO
  – Service Catalog used for app-level services
  – Endpoint Filtering used
  – Inherits support for FIM, PKI, certificate caching, revocation lists, etc.

• New rule set enforces three pre-defined roles:
  – voms_admin
  – vo_admin
  – vo_site_admin

• Modular VO Policy Enforcement Point built
  – Based on WSGI

A KeyVOMS Example: Managing RSS Feeds

- KeyVOMS CLI client (python-based) used to interact with KeyVOMS in all roles:
  - voms_admin, vo_admin, vo_site_admin, and ordinary users
- RSS_VO domain created with Member-A in Project-A and Member-B in Project-B
- An RSS Feeder (python feedparser, v 5.1.3) integrated with WSGI-based VO PEP:
  - The “tech” RSS topic (endpoint) assoc’d with Project-A; “middle_east” topic w/ Project-B
- Total KeyVOMS Service Catalog:

<table>
<thead>
<tr>
<th>Svc ID</th>
<th>Svc Type</th>
<th>Endp ID</th>
<th>Endp I/F</th>
<th>Endp Region</th>
<th>Endp URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;svc_id1&gt;</td>
<td>identity</td>
<td>&lt;endp_id1&gt;</td>
<td>admin</td>
<td>regionOne</td>
<td><a href="http://54.81.20.242:35357/v3">http://54.81.20.242:35357/v3</a></td>
</tr>
<tr>
<td>&lt;svc_id2&gt;</td>
<td>RSS</td>
<td>&lt;endp_id2&gt;</td>
<td>public</td>
<td>None</td>
<td><a href="http://174.129.171.253:6543/tech">http://174.129.171.253:6543/tech</a></td>
</tr>
<tr>
<td>&lt;svc_id3&gt;</td>
<td>RSS</td>
<td>&lt;endp_id3&gt;</td>
<td>public</td>
<td>None</td>
<td><a href="http://174.129.171.253:6543/middle_east">http://174.129.171.253:6543/middle_east</a></td>
</tr>
</tbody>
</table>

- Service Catalog returned to User-A on successful VO login:

<table>
<thead>
<tr>
<th>Svc ID</th>
<th>Svc Type</th>
<th>Endp ID</th>
<th>Endp I/F</th>
<th>Endp Region</th>
<th>Endp URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;svc_id2&gt;</td>
<td>RSS</td>
<td>&lt;endp_id2&gt;</td>
<td>public</td>
<td>None</td>
<td><a href="http://174.129.171.253:6543/tech">http://174.129.171.253:6543/tech</a></td>
</tr>
</tbody>
</table>

- Headline that User-A got from NY Times RSS feed:
  - “Hackers set up fake news organization”

A KeyVOMS Proxy Architecture Diagram

Resource service does not need to be modified with VO PEP, but at the cost of extra overhead and latency.
Design Issues for a P2P Keystone

Client-Side: *Federated Authentication & Service Discovery*

Server-Side: *Federated Credential Validation & Authorization*
Distributed P2P VOMS Design Options

Fed AuthN & Service Discovery

1. Keystone A queries all other Keystones participating in VO Foo. The SC returned contains all possible VO services that the User is authorized to use.

2. Rather than return an SC with all possible service endpoints, the SC contains only the AUTH URL of all other Keystone participating in the given VO. The User can then query the different Keystones on-demand to discover the VO services at each site.

3. Rather than returning an SC with the AUTH URLs for all possible sites, the SC may contain only those AUTH URLs that are known to the User's local Keystone. Each Keystone, then, would support a discovery service, whereby the User could discover additional participating Keystones. The User could, if desired, attempt to discover the closure of all possible sites, but could also choose just to find a suitable subset.

Fed Credential Validation & AuthZ

1. The SP knows which Keystone to use for token validation. The Service Owner must delegate token validation to the remote Keystone.

2. The User's local Keystone provides an auth token for each site represented in the SC returned to the user. This requires the allocation of many tokens upfront, and the user must know when to use which.

3. User uses their Keystone A credentials to obtain VO credentials from Keystone B. Keystone B must validate User with Keystone A. User must obtain credentials from every VO site.

4. Service validates with local Keystone B. Keystone B recognizes the credentials as issued by Keystone A and validates accordingly. User needs only one set of VO credentials. VO authorization can take longer.

5. User’s local Keystone acts as a proxy to the remote site. It authenticates to the remote Keystone and then calls the service on behalf of the local user. Hence, the user delegates its identity and authorization attributes to the local Keystone. This reduces the number of issued tokens and trust relationships, at the cost of increasing latency by going through a proxy.
Local Keystone discovers all service endpoints available across all federated sites and returns them to the user.
Federated AuthN & Service Discovery Option #2

Local Keystone discovers the AUTH_URL of all federated sites and returns them to the user. The user must then discover the relevant service endpoints at each site.
Federated AuthN & Service Discovery Option #3

Local Keystone returns just one AUTH_URL of a federated site. User must then discover other federated sites through the known site, along with their relevant service endpoints.
Federated Credential Validation & AuthZ Option #1

PEP is statically configured with “out-of-band” information just to know where to validate credentials.
Federated Credential Validation & AuthZ Option #2

User is provided a Keystone 2 Auth Token as part of federated authentication. PEP validates w/ local Keystone as usual.
1-4: User uses their Keystone 1 credential to obtain Keystone 2 credential.
2-3: B validates request with A.
6: PEP validates w/ local Keystone as usual.
Federated Credential Validation & AuthZ Option #5

Keystones act as proxies for all interactions. User and SP only interact with their local Keystones.
A General Implementation Concept: *Federation Agents*

- A Federation Agent is the thing that is capable of managing a local user's interactions with a federation:
  - Federated Identity Management & Authentication
  - Resource Discovery
  - Federated Credential Validation
  - Authorization
- A Federation Agent (FA) can be either:
  - Internal to the User's administrative domain, or
  - External to the User's administrative domain
Federation Agent Use Cases

Simple, Pair-wise Federation

- Keystone
- FA
- Keystone

Centralized, Trusted Third-Party

- Keystone
- FA
- Keystone

P2P Federation

- Keystone
- FA
- Keystone

Gateway Federation

- Keystone
- FA
- Keystone

26
A Federated Cloud Concept Map Using FAs

A Work in Progress

Depending on

- Weak vs. Strong Federation
- Bounded vs. Unbounded Federation
- Int. vs. Ex. Federation

Agents all "models of federation" in this design review "fall out" of this concept map.
One Last Topic: Trust Federations

• Trust relationships and common semantic understandings should be established ahead of need
  – Roles, attributes, etc. – *Attribute Name Space Design*
  – VO governance agreements

• Precedent: Global computing grids enabled by the Interoperable Global Trust Federation (IGTF), [www.igtf.net](http://www.igtf.net)
  – Enables trust by defining and auditing PKI Certificate Authority operation
Final Observations

• Keystone is very close to what is needed for general federation management
  – Opportunity exists for Keystone to facilitate a truly global Intercloud

• Many aspects to the Federation Design Space
  – Centralized, third-parties
  – Proxies
  – Decentralized -- peer-to-peer vs. gateways
  – Scale: from simple pair-wise federations to a global Intercloud

• There is no "best" federation approach
  – Different approaches will be appropriate in different application scenarios

• Much more experience and socialization is needed!
  – Testbeds, Testbeds, Testbeds:  *IEEE Intercloud Testbed Project!*
  – Apps, Apps, Apps!
  – Demos, Demos, Demos!