Access Control Venn Infographics

UMA Work Group

16 July 2013
Comparing three technologies

- OpenID Connect
- UMA
- OAuth 2.0
OpenID Connect
Profiles as a claims-gathering option

Profiles for SSO API protection

Profiles to solve access management

OAuth 2.0

Their relationships
OAuth in a nutshell (as usually deployed)

- You delegate scope-constrained access to other apps
- Apps can get access using a variety of token types
- You grant access to apps operated by you
- You control access to web APIs
- Calling app is recognized based on authenticated identity
- Apps can get access after you go offline
- You grant access by consenting to terms at run time
- The authorization function is effectively local to resources

OAuth 2.0
OpenID Connect in a nutshell

- You achieve federated **single sign-on** and login-time attribute exchange.
- You control access to **claims about you**.
- Apps get access using **bearer-style tokens**.
- You delegate **scope-constrained** access to other apps.
- You grant access by **consenting** to terms at run time.
- You grant access to apps operated by **you**.
- The authorization function is effectively **local** to resources.
- Calls app is recognized based on **authenticated identity**.
- Apps can get access after you go **offline**.
- **Claims** can come from distributed sources.
What OAuth and OpenID Connect share

**OpenID Connect**
- You achieve federated **single sign-on** and login-time attribute exchange
- You control access to **claims about you**
- Apps get access using **bearer-style tokens**
- You grant access by **consenting** to terms at run time
- You grant access to apps operated by **you**
- The authorization function is effectively **local** to resources

**Claims** can come from distributed sources

**OAuth 2.0**
- You delegate **scope-constrained** access to other apps
- Calling app is recognized based on **authenticated identity**
- Apps can get access after you go **offline**
- Apps can get access using a **variety** of token types
- You control access to **web APIs**

Profiles for SSO API protection

What OAuth and OpenID Connect share
UMA in a nutshell

- **Claims** can come from distributed sources
- You **delegate scope-constrained** access to other apps
- Calling app is recognized based on **authenticated identity**
- Apps can get access after you go **offline**
- You can grant access to apps operated by **anyone**
- You can control access to **any** type of web resource
- You can grant access by **setting policies and terms** ahead of time
- The authorization function is **standard and centralizable**
- Requesting party is authorized based on **claims**
- Apps can get access using a **variety** of token types
What OAuth and UMA share

**OAuth 2.0**
- You control access to web APIs
- The authorization function is effectively local to resources
- You grant access to apps operated by you
- You grant access by consenting to terms at run time
- You can grant access by setting policies and terms ahead of time
- Requesting party is authorized based on authenticated identity
- The authorization function is standard and centralizable
- Profiles to solve access management

**UMA**
- You can grant access to apps operated by anyone
- You can control access to any type of web resource
- You can grant access by setting policies and terms ahead of time
- The authorization function is standard and centralizable
- Requesting party is authorized based on claims
- Apps can get access using a variety of token types
- Apps can get access after you go offline
- Claims can come from distributed sources

**What OAuth and UMA share**
- You delegate scope-constrained access to other apps
What OpenID Connect and UMA share

OpenID Connect

- You achieve federated **single sign-on** and login-time attribute exchange
- You control access to **claims about you**
- Apps get access using **bearer**-style tokens
- You grant access by **consenting** to terms at run time
- You grant access to apps operated by **you**
- The authorization function is effectively **local** to resources

UMA

- **Claims** can come from distributed sources
- You **delegate scope-constrained** access to other apps
- Calling app is recognized based on **authenticated identity**
- Apps can get access after you go **offline**

Profiles as a claims-gathering option

- You can grant access to apps operated by **anyone**
- You can control access to **any** type of web resource
- You can grant access by **setting policies and terms** ahead of time
- The authorization function is **standard and centralizable**
- Requesting party is authorized based on **claims**
- Apps can get access using a **variety** of token types
Controlling access: by what/whom?

- **OpenID Connect**: You grant access to apps operated by you.
- **OAuth 2.0**: Calling app is recognized based on authenticated identity.
- **UMA**: Requesting party is authorized based on claims.
- **UMA**: You can grant access to apps operated by anyone.
Controlling access: to what?

- **OpenID Connect**
  - You control access to claims about you

- **UMA**
  - You can control access to any type of web resource

- **OAuth 2.0**
  - You control access to web APIs
Controlling access: by what means?

You grant access by consenting to terms at run time.

You can grant access by setting policies and terms ahead of time.

- OpenID Connect
- UMA
- OAuth 2.0
Authorization function: how is it coupled?

- OpenID Connect
  - The authorization function is effectively local to resources
- UMA
  - The authorization function is standard and centralizable
- OAuth 2.0
You delegate scope-constrained access to other apps.

OpenID Connect

- You achieve federated single sign-on and login-time attribute exchange.
- You control access to claims about you.
- Apps get access using bearer-style tokens.
- You grant access by consenting to terms at run time.
- The authorization function is effectively local to resources.

UMA

- You can grant access to apps operated by anyone.
- You can control access to any type of web resource.
- You can grant access by setting policies and terms ahead of time.
- The authorization function is standard and centralizable.
- Requesting party is authorized based on claims.

OAuth 2.0

- You control access to web APIs.
- Apps get access using a variety of token types.
- Apps can get access after you go offline.
- The calling app is recognized based on authenticated identity.
- You grant access by consenting to terms prior to use.

Profiles as a claims-gathering option.

Profiles for SSO API protection.

Profiles to solve access management.

Summary