Challenges in apps that handle personal data and content
Some apps are still in the Web 1.0 dark ages

- Provisioning user data by hand
- Provisioning it by value
- Oversharing
- Lying!
Some other apps are still in the Web 2.0 dark ages

- The “password anti-pattern” – a third party impersonates the user
- It’s a shared secret honeypot
- It’s a gray-market B2B partner
Apps using OAuth and OpenID Connect hint at a better, if not perfect, way
Apps using OAuth and OpenID Connect hint at a better, if not perfect, way
What about selective person-to-person sharing?
Our choices: send a private URL...

- Handy but insecure
- Unsuitable for really sensitive data
Or implement a proprietary access management system...
Or require impersonation

Import Fidelity Tax Information Into TurboTax®

If you are a Fidelity customer and use TurboTax®, you may be able to import certain information directly from your account into the software. Here’s how.

How to import your information

Once you receive your 1099 statement by mail or through eDelivery, have it available to verify the imported information. Follow these simple steps:

1. Enter your Social Security number (SSN), taxpayer identification number (TIN), or username, and then your password. When asked where to import information from, select Fidelity Investments and enter the same information that you use to log on to Fidelity.com. Then, the tax information available for each of the accounts associated with your SSN should appear.
Killing – or even **wounding** – the password kills impersonation
We have tough requirements for delegated authorization

- Lightweight for developers
- Robustly secure
- Privacy-enhancing
- Internet-scalable
- Multi-party
- Enables end-user convenience
Introducing UMA
UMA in a nutshell

- It’s a draft standard for “authorization V.next”
- It’s a profile and application of OAuth V2.0
- It’s a set of authorization, privacy, and consent APIs
- It’s a Work Group of the Kantara Initiative
- It’s not an “XACML killer”
- Founder, chair, and “chief UMAntarian”:
  - It’s heading to V1.0 in Q1 2015
The new Venn of access control

OpenID Connect

UMA

OAuth 2.0

FORGEROCK™

15
The UMA protocol enables key new use-case options

I want to **share** this stuff selectively
- Among my own apps
- With family and friends
- With organizations

I want to **protect** this stuff from being seen by everyone in the world

I want to **control** access proactively, not just feel forced to consent at run time
UMA is about interoperable, RESTful authorization-as-a-service
UMA is about interoperable, RESTful authorization-as-a-service

Outsources protection to a centralizable authorization/consent server

Has standardized APIs for privacy and “selective sharing”
UMA is about interoperable, RESTful authorization-as-a-service

- Has standardized APIs for privacy and “selective sharing”
- Outsources protection to a centralizable authorization/consent server

Diagram:
- Resource owner
- Authorization server
- Client
- "Authz relying party" (AzRP)
- SSO relying party (RP)

Outsourcing to a centralizable authorization/consent server

Has standardized APIs for privacy and “selective sharing”
UMA is about interoperable, RESTful authorization-as-a-service

Outsources protection to a centralizable authorization/consent server

Has standardized APIs for privacy and "selective sharing"
Use-case scenario domains

Health
Financial
Education
Personal
Government
Media
Behavioral
Use-case scenario domains

- Health
- Financial
- Education
- Personal
- Government
- Media
- Behavioral

Technologies:
- Web
- Mobile
- API
- IoT
UMA-enabled systems can respect policies such as…

Only let my tax preparer with email TP1234@gmail.com and using client app TaxThis access my bank account data if they have authenticated strongly, and not after tax season is over.

Let my health aggregation app, my doctor’s office client app, and the client for my husband’s employer’s insurance plan (which covers me) get access to my wifi-enabled scale API and my fitness wearable API to read the results they generate.

When a person driving a vehicle with an unknown ID comes into contact with my Solar Freakin’ Driveway, alert me and require my access approval.
The user experience can simulate OAuth or proprietary sharing paradigms, or even be invisible ("better than OAuth")
Under the hood, it’s “OAuth++”

Loosely coupled to enable an AS to onboard multiple RS’s, residing in any security domains

This concept is new, to enable person-to-person sharing driven by RO policy vs. runtime consent
The UMA consent model supports robustly partitioned rights and obligations.
The UMA consent model supports robustly partitioned rights and obligations.

- **Authorizing Party**
- **Resource Server Operator**
- **Authorization Server Operator**
- **Requesting Party**

Underlies access federation trust frameworks.

Akin to...

Identity federation trust frameworks.
Thank you!

Eve Maler
VP Innovation & Emerging Technology
eve.maler@forgerock.com
@xmlgrrl
Appendix:
The gory UMA details
The RS exposes whatever value-add API it wants, protected by an AS.

The RPT is the main "access token" and (by default – it's profitable) is associated with time-limited, scoped permissions.
The AS exposes an UMA-standardized protection API to the RS

The PAT protects the API and binds the RO, RS, and AS
The AS exposes an UMA-standardized authorization API to the client.

The AAT protects the API and binds the RqP, client, and AS.

The client may be told: "need_claims"
The AS can collect requesting party claims to assess policy

A “claims-aware” client can proactively push an OpenID Connect ID token, a SAML assertion, a SCIM record, or other available user data to the AS per the access federation’s trust framework.

A “claims-unaware” client can, at minimum, redirect the requesting party to the AS to log in, press an “I Agree” button, fill in a form, follow a NASCAR for federated login, etc.