The “data price” for online service is too high: typing…

- Provisioning by hand
- Provisioning by value
- Oversharing
- Lying!
The “data price” for online service is too high: connecting...

- Meaningless consent to unfavorable terms
- Painful, inconsistent, and messy access management
- Oblivious oversharing
The “data price” for online service is too high: private URLs...

- Handy but insecure
- Unsuitable for really sensitive data
Most data “sharing” today is back-channel and unconsented
Privacy is about context, control, choice and respect – so UMA enables a “digital footprint control console”

• Web 2.0 access control is inconsistent and unsophisticated
• To share with others, you have to list them literally
• You have to keep rebuilding your “circles” in new apps
• You can’t advertise content without giving it away
• You can’t get a global view of who accessed what

• You can **unify** access control under a single app
• Your access policies can test for **claims** like “over 18”
• You can **reuse** the same policies with multiple sites
• You can control access to stuff with **public** URLs
• You can manage and **revoke** access from one place
Enterprise use cases bring WAM into the API economy

- Scopes are entirely proprietary and non-interoperable
- Access management and policies are done on a pairwise, per-service basis
- You create and standardize machine-readable scope descriptions
- You can centralize scope mgmt at one AS and reuse policies

- The RO is the enterprise itself
- The policy administrator is an “RO agent”
- The AS is a PAP and (pseudo) PDP that can serve as a PIP client
Protocol vs. value-add: the basics
ASSUMPTION: STILL HAS API-SPECIFIC SEMANTICS, JUST LIKE OAuth

- Apps can outsource reusable high-quality access control
- Your access policies can test for claims like “over 18”
- You can delegate constrained access to autonomous others
- You can control access to stuff with public URLs
- You can manage and revoke access from one place
- You create and standardize machine-readable scope descriptions
- You can centralize scope mgmt at one AS and reuse policies

- Protocol + likely AS/RS agreements
- Protocol + policy/claim support in AS UX and functionality
- Protocol + policy/claim support in AS UX and functionality
- Protocol + “personal discovery” features
- AS UX and functionality
- Profiling
- Protocol
Potential ecosystem: “social access control” (à la social sign-in)

Most dynamic; Alice-to-Bob sharing is the key differentiator

Benefits
• High-quality, centralized consumer authz

Challenges
• Disruptive change to biz models
• Trust and assurance
• API interoperability

- Few, large, IdP-assoc/PDS
- Some with onboard RS apps

- Work with popular AS+IdPs
- May outsource local authz

- Third-party apps UMA-enabled
Potential ecosystem: “walled garden PDS’s”

Likely highly static partnerships; Alice-to-Alice/Bob/org sharing

Benefits
• Today’s back-channel user data is put under user control/monitoring
• “Outward” trust growth

Challenges
• Tight binding to the owner of the garden

- NSTIC-ish banks and telcos
- In-house apps
- Part of existing third-party ecosystem
- Few truly independent apps
Potential ecosystem: “patient-centric health vaults”

Static partnering will center on payers as 900-lb gorillas; highly vertical

Benefits
• Proactive, trackable consent directives
• Blue Button-like delivery of data

Challenges
• Sclerotic IT practices
• Serious security, privacy, and discoverability needs
Potential ecosystem: “distributed authz for business” (access management 2.0)

AliceCo-to-Employee/Contractor/PartnerBob sharing

- AS
- AS
- AS C
- AS RS

- RS
- RS C
- RS
- RS C

- C
- C
- C
- C
- C
- C
- C

Benefits
- Centralized scope mgmt across web, mobile
- Less dependent on a “big bang”

Challenges
- Legacy apps and WAM practices

- Firms have own AS, like IdP
- May have internal apps

- SaaS, PaaS, IaaS
- “Claims-based SSO”

- Third-party apps UMA-enabled
UMA turns online sharing into a privacy-by-design solution
UMA turns online sharing into a privacy-by-design solution
 Uma turns online sharing into a privacy-by-design solution

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
UMA turns online sharing into a privacy-by-design solution

I want to control access proactively, not just feel forced to consent over and over
Key use cases

http://kantarainitiative.org/confluence/display/uma/Case+Studies

- Subscribing to a friend’s personal cloud
- Sharing accessibility attributes (“GPII”)
- E-transcript sharing (“HEAR”)
- Patient-centric health data access
- Enterprise “access management 2.0”
UMA is a profile of OAuth, with bits added for interop and scale.
UMA solves for
1) individual choice and
2) fully modular cloud services
UMA solves for 1) individual choice and 2) fully modular cloud services.
UMA solves for
1) individual choice and
2) fully modular cloud services

authorization API token

Authorization API token supports OpenID Connect-based claims-gathering for authz
Key implementations

http://kantarainitiative.org/confluence/display/uma/UMA+Implementations

• SMARTAM.net (running authorization service from Cloud Identity UK)
• Puma (Python libraries for RS- and client-enabling web apps) from ditto
• Fraunhofer AISEC open-source implementation in Java
• Gluu OX open-source implementation for Access Management 2.0 use cases
Next steps

• Work on optimization opportunities when UMA and OpenID Connect are used together
• Issue “Implementor’s Draft”
• Continue to work with AXN, Scalable Privacy, and others in “trusted identities in cyberspace” ecosystem
• Profile UMA for higher ed, accessibility attribute sharing, healthcare use cases
• We welcome your involvement and contributions
  – Become an UMAitarian!
  – Follow @UMAWG on Twitter and UserManagedAccess on FB
Questions?
Thank you

@UMAWG
tinyurl.com/umawg | tinyurl.com/umafaq
IIW 16, May 2013
Phase 1: protect a resource

UMA phase 1: protecting a resource (rev 07b)

Section references are from http://docs.kantarainitiative.org/uma/draft-uma-core.html dated 6 Jan 2013

Token terminology:
* PAT = protection API token

Binding obligations terminology, as shown in notes over entities representing obligated parties (see http://docs.kantarainitiative.org/uma/draft-uma-trust.html):
* Subject = Individual or Non-Person Entity
* Authorizing Party = Subject acting as resource owner
* AS Operator = Subject operating authorization server endpoint
* RS Operator = Subject operating resource server endpoint

Learn AS location out of band (Sec 2)

Look up AS config data (Sec 1.5)

AS config data (Sec 1.5)

Dynamic client registration if necessary (Sec 2, draft-ietf-oauth-dyn-reg)

Get PAT using embedded OAuth flow (authorization code grant flow shown) (Sec 1.3.1)

Redirect to AS...

...to log in and consent to PAT issuance

Issue PAT

RS Operator-Authorizing Party: Delegate-Protection

RS Operator-AS Operator: Register-Accurately-and-Timely

Authorizing Party-AS Operator: Introduce-Resource-Server


Register resource sets (Sec 2, draft-hardjono-oauth-resource-reg)

Register resource sets, presenting PAT

Confirm registration

AS Operator-Authorizing Party: Follow-Policies-Accurately-and-Timely
Phases 2 and 3: get authorization and access resource.

Section references are from http://docs.kantarainitiative.org/uma/draft-uma-core.html dated 6 Jan 2013.

Token terminology:
* PAT = Protection API token
* AAT = Authorization API token
* RPT = Requesting Party token

Binding obligations terminology, as shown in notes over entities representing obligated parties (see http://docs.kantarainitiative.org/uma/draft-uma-trust.html):
* Subject = Individual or Non-Person Entity
* Authorizing Party = Subject acting as resource owner
* AS Operator = Subject operating authorization server endpoint
* RS Operator = Subject operating resource server endpoint
* Requesting Party = Subject acting as requesting party

Flow scenario:
* Client starts out with no AAT or RPT but is ultimately able to qualify for the required authorization

Client presents no RPT (Sec 3.1.1)

Learn protected resource location and scopes out of band (Sec 3.1)

Attempt access with no RPT

401 with AS location

Look up AS config data (Sec 1.5)

AS config data (Sec 1.5)

Dynamic client registration if necessary

(Sec 3.4, draft-ietf-oauth-dyn-reg)
Phases 2 and 3: get authorization and access resource

2 of 3
Phases 2 and 3: get authorization and access resource

1 of 3
Spec call tree for the UMA profile of OAuth

- UMA core
- OAuth 2
- OpenID Connect
- Token introspection
- OAuth resource set registration
- UMA binding obligations
- Dynamic client registration
- hostmeta

UMA native spec
Required external component
Optional external component
Individual IETF I-D