The ProtectServe and Relationship Manager Proposition

Eve Maler
Sun Microsystems, Inc.
eve.maler@sun.com
www.xmlgrrl.com/blog/categories/protectserve

IIW8 - 19 May 2009
The Venn of human-to-application relationships
The typical VRM data-sharing problem

disclose

site that consumes data
Very high-level requirements/constraints for usability, empowerment, privacy, commerce

• Don’t depend on login time for routine consent gathering
• Loosely couple data setting and data getting
• Allow people to dictate data-sharing and data-usage terms
  > This is the tough one to “sell”!
The usual federated-identity solution
The OAuth solution

communicate 

consumer

store

authorize

service provider

disclose
More specific functional requirements

• Support the notion of a "relationship management" service that an individual can access as an interface mode separate from authenticating to networked applications

• Allow an individual to select policies and enforceable contract terms that govern the granting of responding-service access to requesting services

• Allow an individual to conduct short-term and long-term management of access relationships, including modifying the conditions of access or terminating the relationship entirely

• Allow an individual to audit and monitor various aspects of access relationships

• Allow requesting services to interact directly with responding services in a fashion guided by policy while an individual is offline, reserving real-time user approval for extraordinary circumstances

• Allow requesting services to interact with multiple responding services associated with the same individual
What a relationship manager can offer
Contract and policy setting
What a relationship manager can offer
Data-sharing and service-access analytics
So, the relationship manager solution
So, the relationship manager solution (in the context of federated identity)
Design principles

• RESTful
  > E.g., no dependency on an Atom-based protocol layer
• Generative
  > Can be hooked up to itself and other things in lots of ways
  > No constraints on types/formats of data shared or contract terms
• PEP/PDP separation
  > A sub-genre of RESTfulness, generativity, and simplicity
• Data sharing late-bound to identifier selection
• Reuse OAuth technology, flows, and concepts
• Efficient for all repetitive protocol steps
• Successful “base” for relationship manager UX above it
Basic goals and use cases motivating ProtectServe’s current design

- Internet-scale (e.g., certain kinds of dynamicism)
- Individual-to-service relationship (e.g., a vendor, a .org, a .gov)
- Both raw retrieval (GET) and more sophisticated data manipulation (POST, PUT, DELETE)
- Calendar-sharing primed our use-case pump
- Primary bucket: “Data Dominatrix”
- Secondary bucket: “Hey, Sailor”

- Lots of emergent use cases: PoCo integration, imposing contract terms on all the parties you deal with, etc.
The RM app vs. the User/AM/SP/Consumer endpoints
The RM app vs. the User/AM/SP/Consumer endpoints
The RM app vs. the User/AM/SP/Consumer endpoints
The RM app vs. the User/AM/SP/Consumer endpoints
A high-level overview of ProtectServe

**Step 0**
User introduces AM and SP

**Step 1**
Consumer registers with AM

**Step 2**
Consumer meets required terms

**Step 3**
AM grants access to resource

User has accounts at all services
User provisions policies at AM
User gives Consumer resource URL
User can consent in real time
AM keeps a log for RM functions
## Some comparisons, FWTW

<table>
<thead>
<tr>
<th></th>
<th>Protect Serve + RM</th>
<th>ISIP</th>
<th>r-cards</th>
<th>OpenID</th>
<th>OAuth</th>
<th>ID-WSF</th>
<th>XACML</th>
<th>Mine!</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>login-time attribute transfer</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>back-channel authz’d access</strong></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td><strong>separate policy decision hub</strong></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td><strong>on-board storage of user data</strong></td>
<td>✓ (opt: RM as SP)</td>
<td>✓ (self-issued)</td>
<td>✓ (self-issued)</td>
<td>✓</td>
<td>✓ (see XACML)</td>
<td>✓ (CARML + Privacy Constraints)</td>
<td>✓ (req)</td>
<td></td>
</tr>
<tr>
<td><strong>user-imposed policy</strong></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓ (see XACML)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>binding of ID(s) to data shared</strong></td>
<td>late</td>
<td>early (usually)</td>
<td>early (usually)</td>
<td>early</td>
<td>late</td>
<td>late</td>
<td>late</td>
<td></td>
</tr>
<tr>
<td><strong>sources of data</strong></td>
<td>arbitrarily multi-sourced</td>
<td>1 (card issuer)</td>
<td>limited (self-issued + 1 managed)</td>
<td>1 (OP)</td>
<td>1 per pairwise svc cxn</td>
<td>arbitrarily multi-sourced</td>
<td>1 (Mine! as aggregator)</td>
<td></td>
</tr>
<tr>
<td><strong>RESTful</strong></td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>co-ownership of shared data</strong></td>
<td>future? (Consumer uses POST)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Next steps for this work

• Further use case development
  > Later today and tomorrow...

• Protocol development in Kantara
  > Coordinating with related communities
  > OAuth, CX, XACML, uApprove, Mine!, Mydex, r-cards...

• Seeking collaboration on implementations
  > OpenSSO’s OAuth extension (imminently)
  > Other open source projects
Thank you! Questions?
Eve Maler
eve.maler@sun.com
www.xmlgrrl.com/blog/categories/protectserve

Many thanks to Paul Bryan, Domenico Catalano, Marc Hadley, Andi Lewis, Bill Smith, and other Sun team members for assistance.