Authentication in the Cloud

Stefan Seelmann

Agenda

- Use Cases
- View Points
- Existing Solutions
- Upcoming Solutions

Use Cases

- End user needs login to a site or service
- End user wants to share access to resources
- Users of enterprise A should use services of enterprise B
- Service C needs to access service D
- Service integration

End-User PoV

- I need to register to each site I want use
- I can't remember all usernames and passwords
- My preferred username is taken
- Login for each site I use
- Smooth services integration
- I won't be tracked

Site Owner PoV

- We have to create another user management
  - Registration, login mask, „forgot password“ mess
- Storing passwords is a huge responsibility
- Less forms for users → higher conversion rate
- We want to integrate other services
- Our service must not rely on external services

Identity Provider PoV

- Wants growing user base
- Track data to make money
- Make other depending on identity service

Existing Solutions

- Own User Management
  - Username / password
- Federated Identity Management + SSO
  - OpenID, SAML, Kerberos
  - „Login with Facebook“, „Login with Twitter“
- Access to Resources
  - OAuth, X.509, Basic Authentication

Username / Password + Basic Auth

- Pros:
  - Easy to implement and easy to understand
  - No dependencies to external services
- Cons:

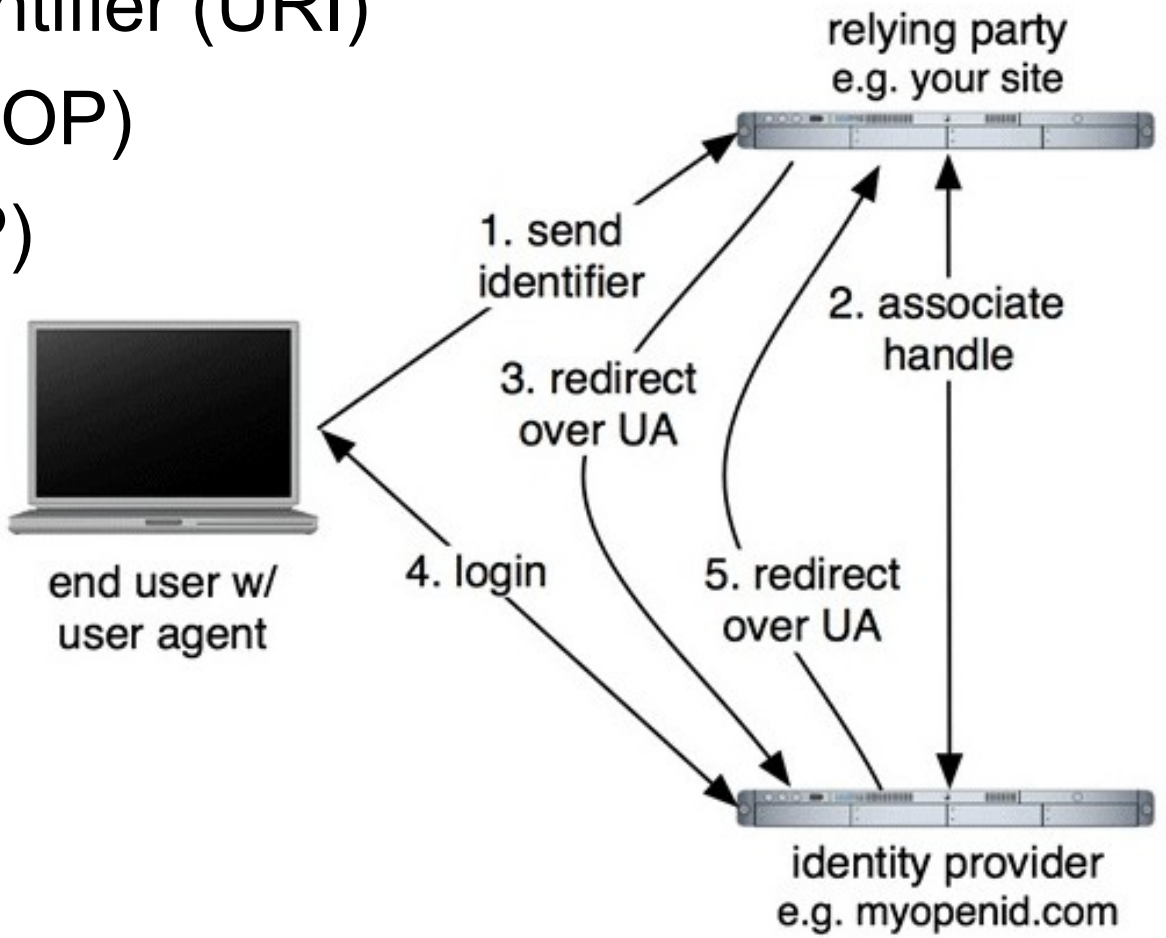
  - Simple passwords, easy to crack
  - Same password for multiple services
  - Disclosed to others
    - When sent over unencrypted connections
    - ftp://user23:w7Xu1$mU@www.example.com/video.mov

OpenID

- Version 2.0 since 2007
- Wants to solve the password problem
- Web Single Sign-On
- Decentral system
  - Ad-hoc addition of RP and OP
- User-centric
  - User selects provider s/he trusts. Or use your own.
  - Delegation

OpenID

- Three Players
  - End user with identifier (URI)
  - Identity Provider (OP)
  - Relying Party (RP)



http://farm1.static.flickr.com/163/351494842_cd83fef2f5_o.jpg

OpenID

- Identifiers:
  - http://exampleuser.livejournal.com/
  - https://www.google.com/accounts/o8/id?id=Abc...
- Delegation
  - http://www.my-domain.com

```
<link rel="openid2.provider openid.server"
      href="http://www.livejournal.com/openid/server.bml"/>
<link rel="openid2.local_id openid.delegate"
      href="http://exampleuser.livejournal.com/"/>
```

OpenID

Adoption?

End User

Site Owner

- Pros:
  - Single identifier, single sign-on
  - Simple registration, attribute exchange
- Cons:

End User

  - OpenID provider may track you
  - Hard to understand (URI as identifier, NASCAR)
  - Doesn't work for mobile apps or JavaScript apps
  - Dependency on external service
    - At runtime; OpenID providers come and go → Delegation

Site Owner

End User

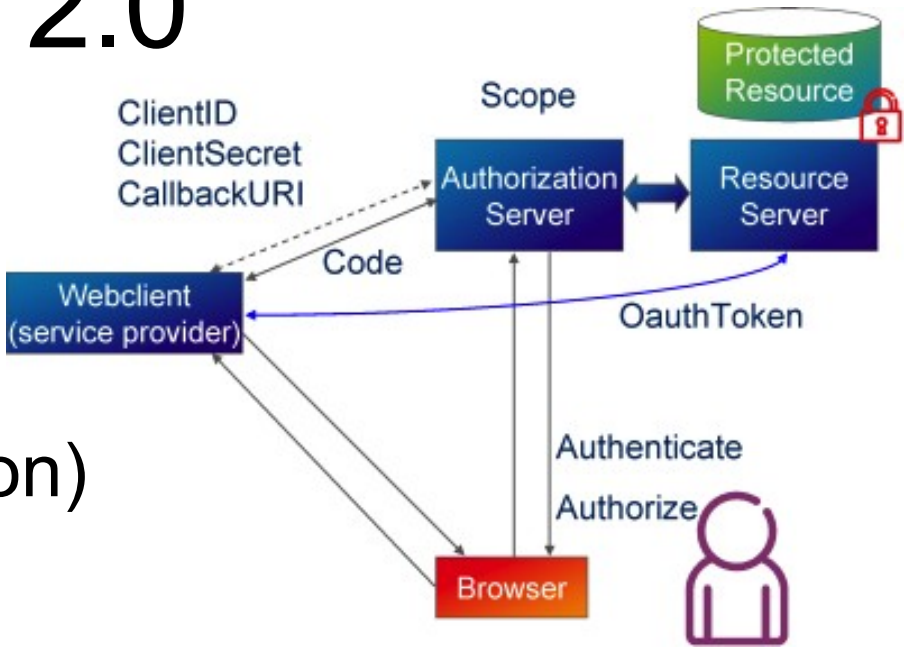
SAML

- Flow similar to OpenID
  - User, service provider (SP), identity provider (IdP)
- Trust between IdP and SP
- XML based
- Used in enterprise and educational environment
- Adoption
  - Implementations: MS ADFS, Shibboleth
  - Google Apps, Salesforge

OAuth

- Let user control 3rd party access to resources
  - Especially to access REST-APIs
  - Default example: photo printing service
- OAuth 1.0 (RFC 5849)
  - Browser-based flow only
  - Crypographic signatures
- OAuth 2.0 (draft 21)
  - Flows for mobile and browser (JavaScript) apps
  - Bearer tokens, SSL/TLS, short-living access tokens

OAuth 2.0



- Four players:
  - Resource owner
  - Client (3rd party application)
  - Authorization server (AS)
  - Resource server (RS) with protected resources
- Tree steps:
  1. Client registration (once per client)
  2. Obtaining authorization (once per user)
  3. Obtaining access token and access resource

https://labs.ericsson.com/apis/oauth2-framework/files/webflow.png


OAuth 2.0

- Client registration
  - Once per client
  - Establish client_id and client_secret
  - Not specified, no protocol


OAuth 2.0

- Obtaining authorization (access/refresh token)
  - Authorization code
    - Server-side webapp
  - Implicit grant
    - JavaScript app
    - Access tokes in URI fragment
  - Resource owner password credentials
    - desktop or mobile app
  - Client credentials
    - if client is resource owner (2-legged)


OAuth 2.0

- Obtaining access token and access resource
  - No more user interaction required
  - Client uses refresh token to obtain access token
    - short-living
  - Client uses access token to access resource
    - resource server must validate access token, not specified

OAuth

Adoption?

End User

- Pros:
  - No need to share credentials
  - Separate access token for each 3rd party
- Cons:
  - OAuth 1.0: complex, no support for apps
  - Flows for mobile and desktop apps still sucks
  - No specification of client registration, AS, RS
  - Requires client registration

Site Owner

Proprietary Stuff

- „Sign in with Twitter“
  - Idea: only the user can allow access
  - OAuth 1.0, registration required
- „Login with Facebook“
  - https://graph.facebook.com/me?access_token=...
  - OAuth2, registration required
- VZ* Login
- Janrain

Upcoming Solutions

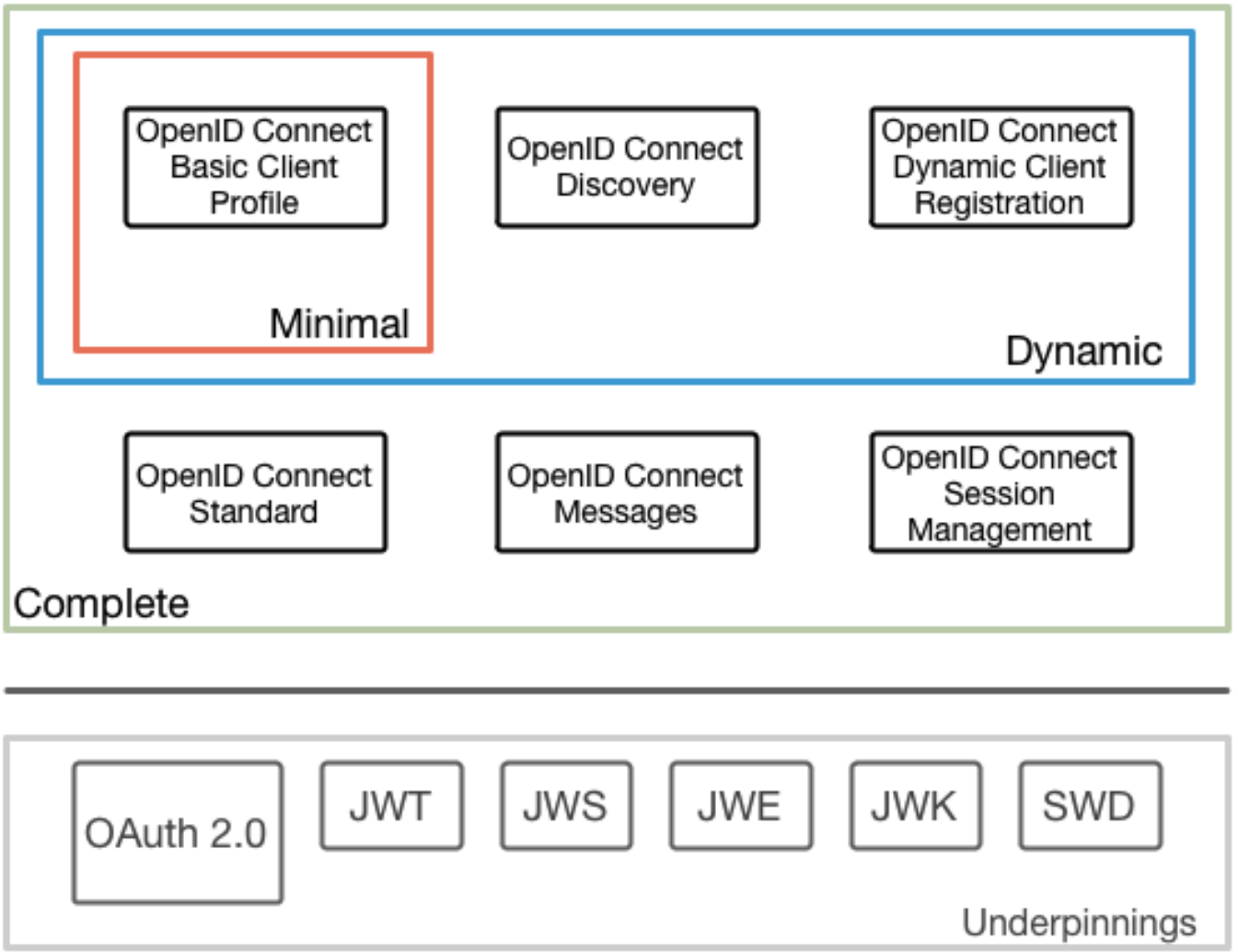
- OpenID Connect
- Account Chooser
- BrowserID
- WebID

OpenID Connect

- Identity service on top of OAuth 2.0
- Login + SSO + access to basic attributess
- Most big players are involved
  - Google
  - Facebook
  - Microsoft
  - Yahoo

OpenID Connect

- Modular



OpenID Connect Protocol Suite

6 September 2011

http://openid.net/openidconnect

OpenID Connect

- Demo
  - https://oauthssodemo.appspot.com

OpenID Connect

- Differences to OpenID 2.0
  - Basics are simpler
    - Attribute exchange (UserInfo endpoint) is built-in
    - No cryptograhic signatures required → HTTPS
    - No library required → HTTP parameters, JSON
  - No more delegation
    - Can be achieved through discovery
  - No longer decentral
    - Dynamic Client Registration is optional
    - Optimized for big players

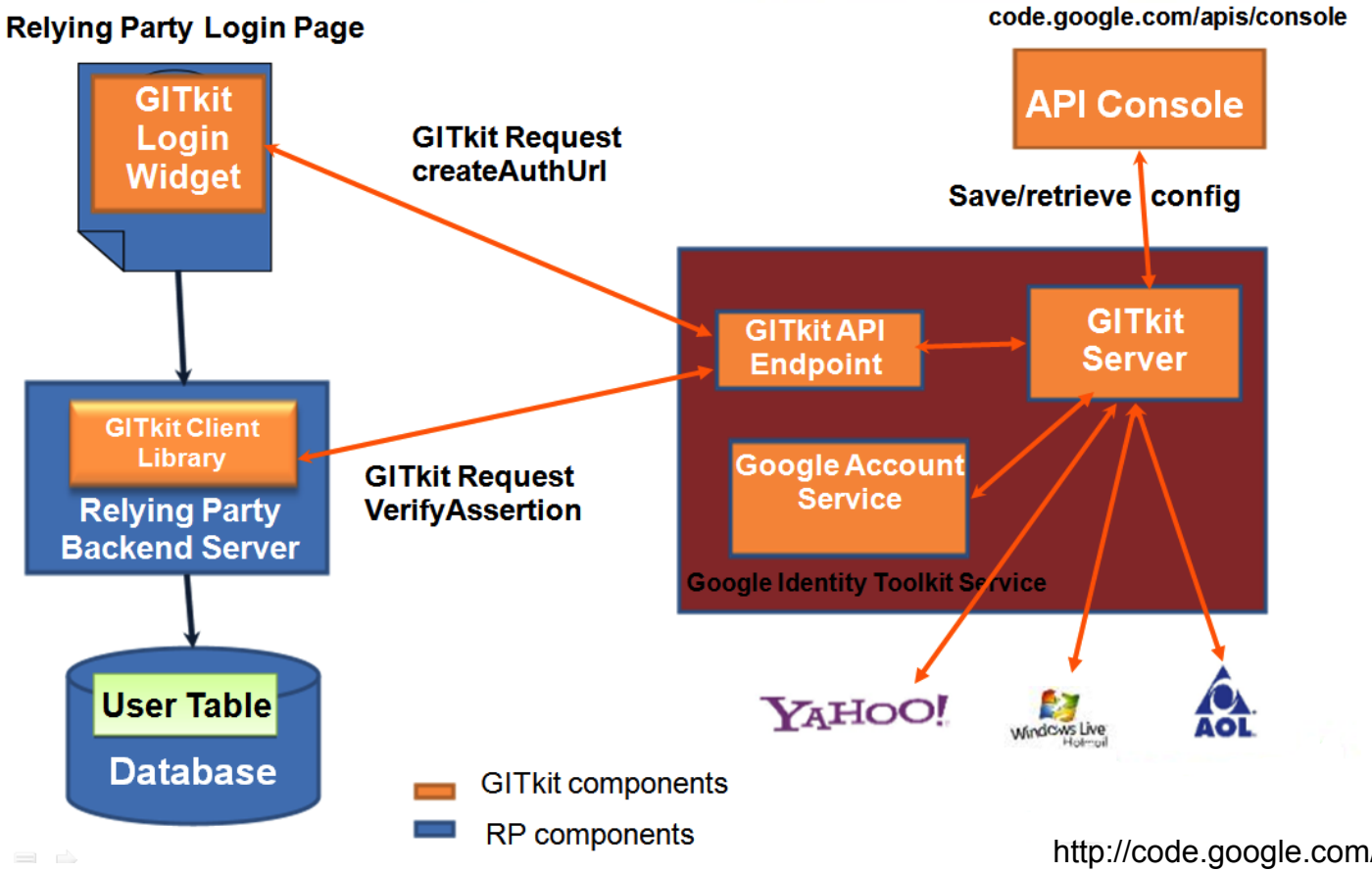
Account Chooser

- Invented by Google
  - Transfer to OpenID Foundation in progress
- Spec: http://accountchooser.com/
- Legacy compatibility (local user database)
- Protocol agnostic (OpenID, SAML, etc.)
- Demos
  - https://account-chooser.appspot.com/
  - http://www.openidsamplestore.com/basic/

sign in

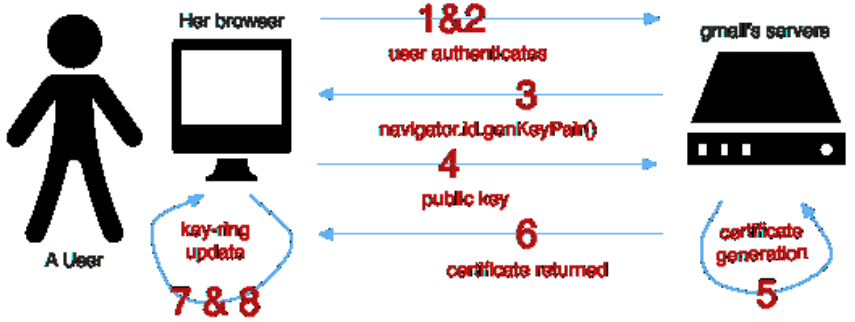
Account Chooser

- Prototype: Google Identity Tookit (GITkit)
  - http://code.google.com/apis/identitytoolkit/index.html



http://code.google.com/apis/identitytoolkit/v1/openid.html

BrowserID



- From Mozilla Labs
- Email is your identity
  - You can verify its ownership
  - Mail provider is primary identity authority
- Implementations:
  - Today: HTML5 based (local storage)
  - Future: natve built into browsers
- Demo: http://myfavoritebeer.org/

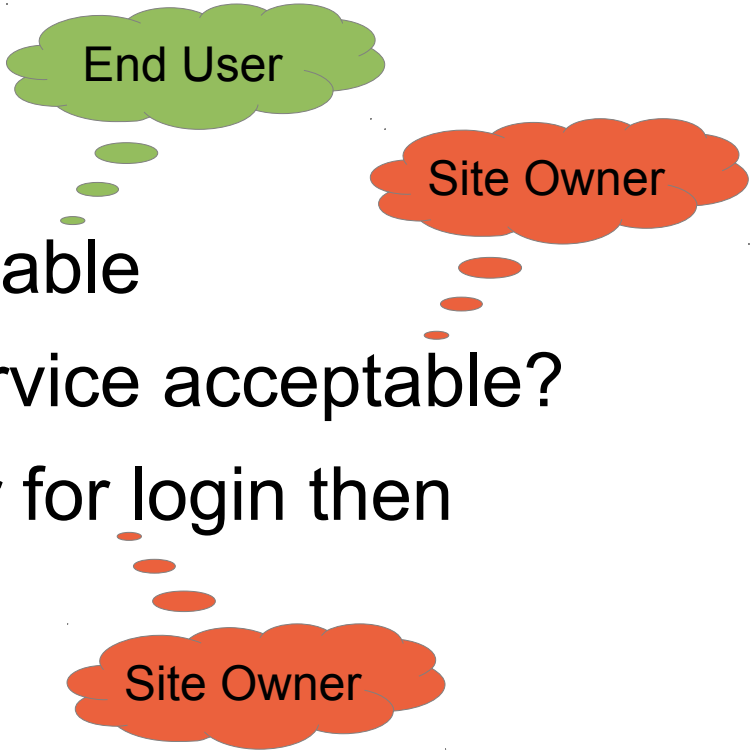
http://lloyd.io/how-browserid-works

WebID

- W3C draft
- Application of Semantic Web
- SSL certificate with pointer to WebID Profile
- WebID Profile is RDF

Summary

- Authentication
  - If you want SSO you are trackable
  - Is dependency on external service acceptable?
  - If you use Facebook of Twitter for login then upgrade to OpenID Connect
- API access
  - OAuth is cool



Resources

- OpenID Connect
  - http://openid.net/connect/
- OAuth 2
  - http://tools.ietf.org/html/draft-ietf-oauth-v2
- Apache Amber (in incubation)
  - Java implemention of OAuth 2.0 (draft 10)
  - Potentially OpenID Connect
  - http://incubator.apache.org/amber
  - https://cwiki.apache.org/confluence/display/AMBER

Thanks for your attention