
Google™
09
I/O

Practical Standards-based Security and Identity

Eric Sachs
Product Manager, Google Security & CIO organization
May 27-28, 2009

“SAML, SaaS, OAuth, OpenID, OpenSocial, .. Oh my!”

Unnamed security "expert"

Part One: Federated Logon (user authentication)

- Especially for enterprise SaaS vendors who target small-medium sized businesses

Part Two: Web service authentication

- Enabling web apps to identify each other across multiple platforms (Windows, Linux, Force.com, App Engine, Gadgets, Azure, Amazon, ...)

Part One: User authentication

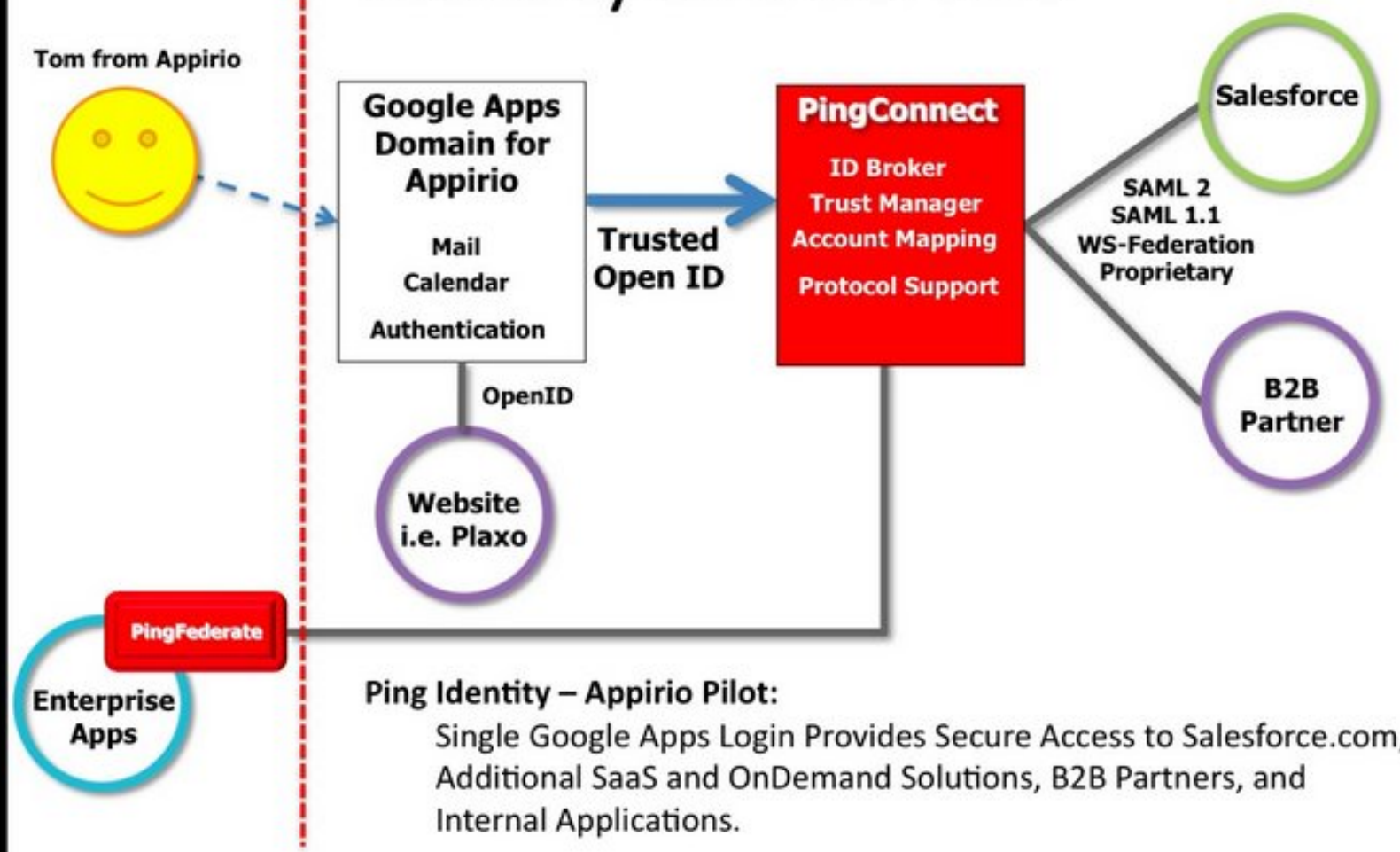
Software IDPs for large enterprises

- Large enterprises
  - Central single-signon (SSO) system
    - Frequently purchased from a software vendor
    - Run on multiple servers and data-centers for high reliability
  - Multiple Software-as-a-Service (SaaS) vendors
    - Salesforce.com
    - Google Apps Premier Edition
    - etc.
  - SAML=open standard for federated login, supported by some SaaS vendors

But what if you are not a large enterprise?

- Small-Medium sized business (SMBs)
  - Option 1: On-premise
    - Usually Active Directory or nothing
    - Don't make SSO a Single Source Of failure!
  - SSO/IDP as a service
    - Growing set of vendors (Ping Connect, Symplified, MS Azure, Google Apps, Lotus Live, ...)

Identity as a Service



Ping Identity – Appirio Pilot:

Single Google Apps Login Provides Secure Access to Salesforce.com, Additional SaaS and OnDemand Solutions, B2B Partners, and Internal Applications.

Guidance for SMBs

Smaller businesses using SaaS offering can similarly use a service based sign-on offering

Quickly evolving space, especially the next few months...

- Chained identity providers
- SAML
- OpenID
- Stronger authentication
- Provisioning
- Group membership
- etc.

Watch for more vendor/service provider interoperability results

What if you are a SaaS vendor?

Demand is changing

- Historically only large enterprises wanted federated login to SaaS vendors.
- Expect growing demand from SMBs

Options

- Roll your own
- Purchase SAML or OpenID relying party software
- Use a RP (relying party) service as a bridge (Janrain, Ping, etc.)

Two challenges to consider

SaaS vendors with native (non-browser) apps

- Most apps are hard-coded to ask for Email/password, and do not work with Federated Login
- Try OAuth

Login box

- Traditional Email/password login box is not sufficient
- Try asking for E-mail address first without a password

...search for "goog oauth" for a site with more details

What if you are a super techie?

Join a standards community!

- OpenID
- OAuth
- SAML
- etc.

Many of the techniques will be shared across those standards.

Lots of work left to do

- Improved user experience
- Automated setup of federated login
- Better crypto hygine
- Integration with provising and group membership

Part Two: Web service authentication

I know how to authenticate a user accessing my app, but how do I authenticate another web-service accessing my app?

Unnamed CIO

Common problems

1. I want to build a frontend app on my Windows server to access a backend hosted on Linux.
2. I want to write a system management console on App Engine to monitor my app running on Amazon (or a different app running on App Engine)
3. I want to build a frontend on App Engine to manage data stored in my ERP system

Welcome the crypto experts

Numerous proprietary and standard options

If API calls route through firewalls, REST API format is frequently the easiest

Try OAuth. The standard is small (spec is only a few pages) and extensible

For more crypto details, visit the Google I/O OAuth helpdesk

Quick version

Traditional REST API call

acme.com/salary?u=sara&s=90000

REST API call + OAuth

acme.com/salary?u=sara&s=90000
&oauth_consumer_key=frontend.acme.com
&oauth_signature=A23F68

consumer_key=claimed name of the calling app
signature=digital signature to prove it came from that app, and that the URL has not been modified

Okay, a little hard

Generating (and verifying) the OAuth signature parameter is the key technique that is standardized by OAuth

Easy to get it wrong, so use open source libraries where possible
http://oauth.net

Crypto hygiene

The signer and verifier need to agree upon the crypto mechanism

1. HMAC (symmetric) - Simper, less CPU intensive
2. RSA (Public Key, asymmetric) - Verifier cannot impersonate the signer

If there are multiple calling apps, issue each of them a different HMAC secret or RSA key pair

I want to build a frontend app on my Windows server to access a backend hosted on Linux.

1. Create a REST (or SOAP) endpoint on the Linux app
2. Make REST calls to it from the Windows app
3. Have the Windows app add oauth_consumer_key with a name that references the Windows app
4. Agree on a crypto scheme
5. Have the Windows app add oauth_signature with a signed version of the REST URL
6. Have the Linux app require a signature on all requests, and verify the signature
7. Confirm the verified calling app is authorized to perform the request (such as changing an employee's salary)

OAuth terminology/history

Basic version: 2-legged OAuth

Original version: 3-legged (consumer scenarios)

Developed in 2007 by a number of consumer oriented websites (Yahoo, AOL, Google, Flickr, Plaxo, Twitter, Digg, ...) to replace similar proprietary mechanisms

Making OAuth even simpler

Some applications run in containers such as an OpenSocial gadget container, App Engine, Google Spreadsheets, etc.

Those containers can perform signing on behalf of apps

Traditional REST API call
acme.com/salary?u=sara&s=90000

Container based REST API call
URLFetch(acme.com/salary?u=sara&s=90000, SIGNED)

Note: function names vary by container. Search for "oauth proxy" for an example

Verifying a "signed-fetch"

Final output from container

acme.com/salary?u=sara&s=90000
&opensocial_app_url=hrtool.appspot.com
&oauth_consumer_key=appspot.com
&oauth_signature=D1C952A

consumer_key=the container
app_url=the app on the container

The container picks the crypto mechanism (including rotation), and the app name. The receiving system just needs to read the container documentation.

I want to write a system management console on App Engine to monitor my app on Amazon

1. Create a REST endpoint on the Amazon app
2. Make URLFetch calls to it from the AppEngine app
3. Have the Amazon app require a signature on all requests, and verify the signature
4. Confirm the verified calling app is authorized to perform the request (such as adding more virtual servers at Amazon)

Search for "oauth google app engine" for software examples.

I want to build a frontend on App Engine to manage data stored in my ERP system

How do I get through my firewall?

VPN, Google Secure Data Connector, ...

Back to our example:

acme.com/salary?u=sara&s=90000

But who wants to change sara's salary?

Best practice:

acme.com/salary?u=sara&s=90000
&opensocial_viewer_email=tom

In this case, Tom is logged into the calling app. Any parameter name can be used, however..

Domain APIs

Standard names improve interoperability

For example

- Company X hosts their email/calendars at Google
- Their admin can register a 2-legged OAuth consumer key and secret with Google
- The company can then run an app to access the calendars of all their employees, for example:

  calendar.google.com/?opensocial_viewer_email=tom

  For more information, attend the Google I/O Session "Building Enterprise applications in the cloud"

Containers adding identity information

If a container is used:
URLFetch(acme.com/salary?u=sara&s=90000, SIGNED)

Final output from container
acme.com/salary?u=sara&s=90000
&opensocial_viewer_email=tom
&opensocial_app_url=hrtool.appsport.com
&oauth_consumer_key=appspot.com
&oauth_signature=D1C952A

Standards under development

Google App Engine

URLVerify(*a signed OAuth URL*)

Three step process

1. Calling app gets secret from Google
2. Use secret to sign urls in external apps
3. App engine can have Google infrastructure verify signature without managing the secret itself

Search for "app engine oauth verify" for software examples.

How simple can we make it?

GAE app 1

URLFetch(acme.com/salary?u=sara&s=90000,SIGNED)

GAE app 2

```
URL=acme.com/salary?u=sara&s=90000
    &opensocial_viewer_id=tom
    &opensocial_app_url=hrtool.appsport.com
    &oauth_consumer_key=appspot.com
    &oauth_signature=D1C952A)
If URLVerify(URL)
    if trustedapp(URL.param(opensocial_app_url,
            oauth_consumer_key)
    if URL.param(u).mananger =
            URL.param(opensocial_viewerid)
    URL.param(u).salary=URL.param(s)
```

Secure mashups are possible!

Questions?

Visit us at the Google I/O OAuth helpdesk staffed by Brian Eaton and Kevin Brown

For more information, use the OAuth mailing list or do a Google search for "oauth goog" for a site with more details


Google™
09
I/O