Implementing CAS

Andrew Petro

open source java application sso for web

applications and users trust CAS, so users log in to applications without presenting passwords to the application

CAS vends and validates short-lived big-random-number tickets (similar to the Kerberos strategy)

What problems does CAS solve? FEW:
- login user experience
- sso
- lightweight attribute release
- n-tier delegated authentication
- selective secure release of password to applications (if necessary)

CAS 3.5 in release process now (June 2012) 3.4.12 is lastest generally available release

CAS sets a secure cookie scoped to CAS;
subsequently CAS recognizes the browser via cookie and does not require re-authN
CAS collects user attributes from other sources at login;
releases attributes to relying applications via SAML 1.1 (not using a profile)
For more rigorous SAML 2 interactions (e.g., federation) recommend bridge to Shibboleth,
which rigorously implements profiles and SAML 2

N-tier authN
one app authenticates to another, without password replay
ex: portal-> web service -> IMAP (cassified to rely on tickets)

Services Registry: designates which services use CAS
so not wide open for any service to attempt to use

CAS 3.5 accepts additional protocols (Oauths, Google accounts or OpenID)

CLIENT LIbraries (CAS a simple protocol, so easy to write these libraries)
- java / spring / apache shiro
- apache httpd mod_auth_cas (relatively straightforward to configure if the app just consumes remote user)
- .net
- php (good for more sophisticated integrations)
- ruby, perl
- drupal, uPortal, Sakai
- simpleSAML PHP (does both CAS and Shibb)

Community - many adopters, active mail lists hosted by JASIG, wiki, conferences

Commercial participants
- separate choice of product from choice of service provider
- commercial providers like unicon

Building and Deploying

Requirements: Java (6 or 7), Maven (2 or 3), Tomcat (7)

download server from JASIG, build using MAVEN, out comes WAR file that will enable demo login (username=password)

CAS Protocol

HTTP GETs
response bodies of lightweight XML

tickets are big random numbers
TGT - authenticaate sbrowser tyo CAS (ticket granting cookie)
service ticket authenticaes browser tyo aspecific relying party
proxy granting ticcket authenticates app to ccas to obtain proxy ticket
proxy ticket authenticates app to another app in context of end-user sso session

Maven Overlay

.pom xml file contains dependencies and other metadata
can keep only local mods and changes locally, indicated in the .pom file; other source from official sources;
modify the deployerConfigContext file, for example, to indicate authentication source such as local LDAP service, the lifetime of tickets, etc.

modify cas.properties file to adjust other properties, including the location of the properties file - put outside the war into context, so can be changed without recompiling.

CASifying Sakai (& others):

1. Config app to trust environment to assert remote user *"container authN")
2. config Java CAS client to authN user using CAS

CASifying uPortal:

uPortal has feature to grab proxy tickets, so portlets can have delegated authN (e.g., for getting email preview without relaying pwd)

Clustering:
need shared ticket registry state across CAS server nodes
options to provide:
  EhCache (preferred) java library built into the war file; configured in the pom.xml
  memcached

CASlogin flow:
detailed interception of the login flow, enabling appropriate error messages, reminders, etc. to enhance user experience
or add second factor...
(Shibb out of the box doesn't provide that)

Columbia U CAS integration

Adoption of GAE: wanted to use CAS login with GAE (CAS already deployed for many apps)
Added a "new protocol" and extended functionality of CAS

Google Apps SAML SSO step-by-step instructions configuring GAS for GAE (jasig wiki)

Migrating the legacy web Auth system ("WIND") cookie-based SSO similar to CAS
Needed skills: java, maven, view technologies (JSP, CSS), Spring configuration, Spring web flow (SWF), app server / web server (tomcat / apache)

Custom service registry: allowed to retain extended service attributes (hence functions) from legacy webAuth:
extends attributes in registeredservice adaptor

Add WIND protocol to CAS: CAS is "essentially" multi-protocol - assumes there will be multiple protocols

Requires extending or replacing several classes within CAS.

Add UI customization from WIND. The login page is modified or added to with application-specific logo and links, if provided.
Write SWF action class to map incoming service to a registeredservice and make registeredservice available in web flow context.

CAS and Shibb - "even more perfect together" Bill Thompson

Enterprise webSSO barriers:
- user experience and expectations
- existing IAM architecture and infrastructure
- enterprise portal
- close source enterprisey systems (Banner etc)
- home-grown technologies
- systems that support one or another protocol
- hard cases: IMIAP, OWA
- federation needs

"CAS is great"
- flexible user interface - SSO session logout, opt in/out SSO per service,
"Shibb is great too"
- SAML, InCommon, cloud-based SAML service providers, LoA, ICAM, NSTIC

Combine: CASify the Shibb server
not supported in this approach: isPassive, forceAuth

Use Shib exterlSuth API to do better bridge: cas_shib_authenticator

Do get gain SSO session logout [sort of…destroy SSO session cookie, but apps themselves not "informed" - unless log out of each app…]

Microsoft ADFS - CAS/SSO

Office 365 requires ADFS. ADFS does support Shibb ("out of the box") - delegates to Shibb for AuthN. Then, of course can further delegate to CAS. Unicon implemented this for Julliard.