X.509 Authentication/Authorization in OpenNebula

Initial goal:

Use all Grid-based X.509 authorization and authentication methods to authenticate and authorize on cloud.

Authentication History:

T. Hesselroth wrote initial X.509 authentication code, submitted to OpenNebula, has been in all OpenNebula since 3.0.

Authentication code:

- Following modifications were made to ONE3.2:
  - FCLogging—add the DN to the logs for:
    - Every “oneuser login” (CLI)
    - Every sunstone login (GUI)
    - Every “onetemplate instantiate” (CLI)
    - Every VM/template launch (Sunstone GUI)
  - Server side for “ECONE” emulation:
    - X509CloudAuth.rb—modify to allow use of proxies
    - EC2queryclient.rb—use libcurl to do X.509 (needed to make all econe-* command line clients work).
- Same modifications (better coded) have been made to ONE 4.4, they work.

Authorization Code

- Goals:
  - Use as much of existing X.509 grid infrastructure as we can.
- History:
- T. Hesselroth wrote ruby xacml adapter module, worked in demo. Decided we didn't want to maintain private library just for this.
- Attempted to use LCMAPS to contact GUMS and SAZ:
  - Used Ruby->C binding
  - Worked on CLI

Problems in browser

- Can load personal cert + proxy into browser
- But browser doesn't recognize personal cert as a valid signing cert, so can't use normal SSL handshake
- LCMAPS doesn't work without full certificate chain passed to it.
- Gridsite should return VO and FQAN for voms-signed proxy but doesn't.
-

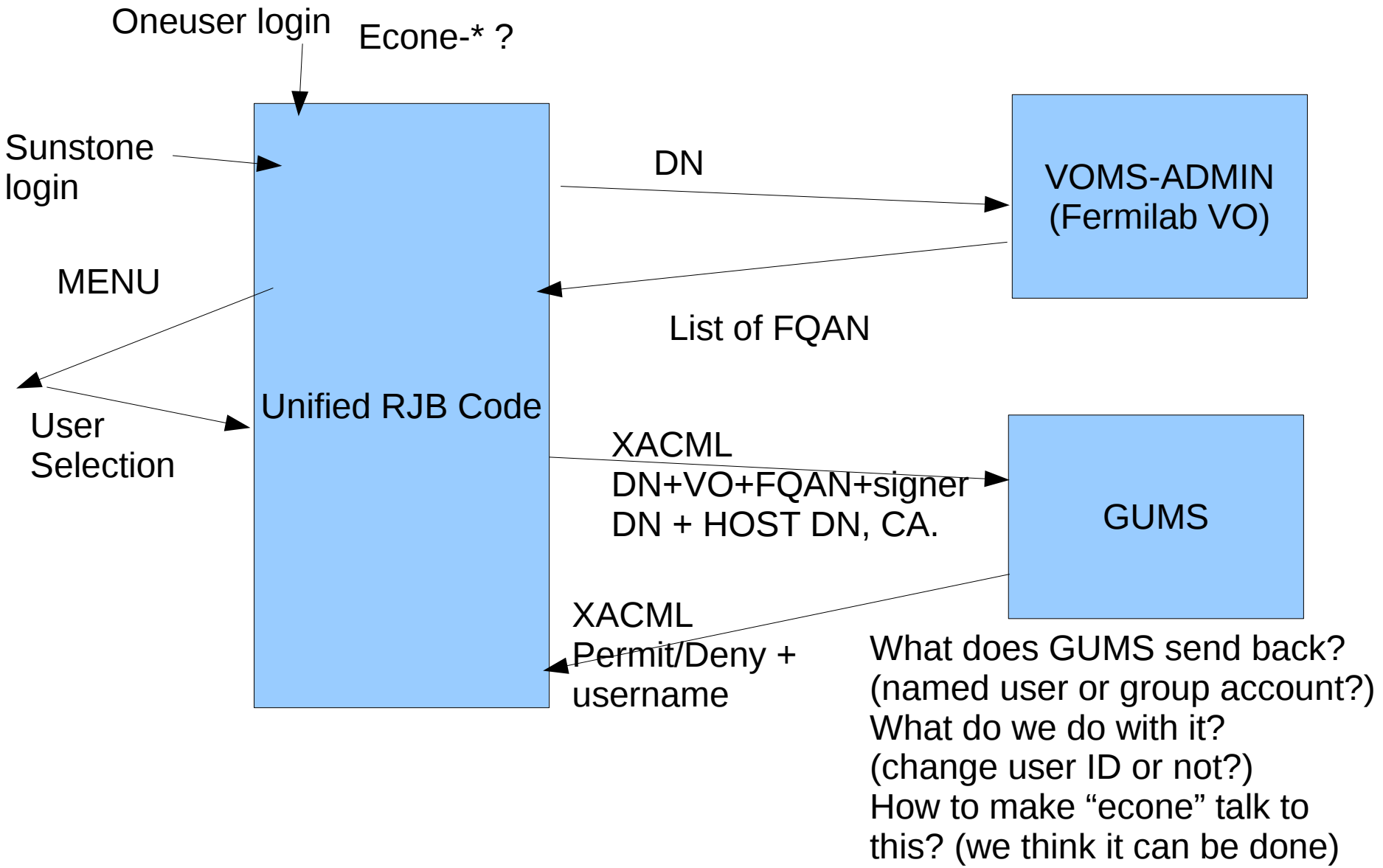
Ruby-Java bridge

- Contact voms-admin to get list of legal VO's that USER DN is part of
- User selects one:
- Then can send DN + FQAN to GUMS, SAZ
- Using code in privilege.jar (which is more flexible in constructing XACML queries than LCMAPS).
- Ruby-java bridge used to invoke the java code.
- Don't need whole certificate + proxy chain this way, just DN

Issues we need to solve:

- Right now DN-to-userID-to-VO is hardwired.
- Need single DN to be able to launch VM's from different VO's.
- Need a way to track which VO the VM belongs to. (in a way that OpenNebula and Gratia can see)
- VO selection process has to work with interactive, in shell script, or via “ECONE”

Single DN, different VO's



Where to store the VO/FQAN info?

- In user template section of VM template?
  - Modifiable by user, but can change permissions so that it is a privileged operations.
  - Is that user template readable by Gratia probe?
- In user context base-64 encoded string?
  - $USER[TEMPLATE]
  - Hard to encode/decode, rest of info changes
  - Is this field readable/modifiable by Gratia probe?
  - In AWS/OpenStack this info available as “instance metadata”
  - GlideinWMS knows how to manipulate, we think.
- As active OpenNebula “group ID” of user?
- Take the “username” returned by GUMS and make it the ONE userid.

GUMS responses

- What does GUMS send back:
  - Permit/Deny decision plus username
  - Could configure “username” to be individual user account or group account.
- Individual user account:
  - Pro—follows current pattern of every individual having his/her own account created on FermiCloud. /fermilab/Role=CloudUser group already exists in cloudgums
  - Con—there is nothing in the user name to track the VO. Would have to do some external way to track VO (see previous slide).
- Group Account
  - Pro—The group account name can easily be associated with VO (can use auto-generated user-vo-map). Your VM runs in the same account as it would on the grid. Also avoids host-specific mapping for FermiCloud.
  - Con—different usage would be blended together and possibly we could end up accepting large number of DN. Hard for FermiCloud operators to tell which DN launched the VM.
- In theory could leverage uid/gid specific GUMS work being done for dCache but FC project management does NOT want to go there for cloud if we could possibly avoid it.

What to do with GUMS responses

- Definitely want to use Permit/Deny especially since SAZ is not in picture
- Username, whatever it is:
  - Make this be the username lookup, in place of mysql database?
  - OR—just store it in a safe location to track VO and use mysql database DN lookup to get ONE username.

Assume this model for planning purposes:

- There will be some way for user to specify FQAN on all FermiCloud interfaces (takes some coding)
- GUMS will return same username for cloud as for grid
- ONE will use this group account as the ONE userid (takes some coding).
- Can use auto-generated user-vo-map from GUMS to map username->VO for Gratia.

Presuming we can successfully answer all questions and make this work:

- Significant effort to make it work
- More to keep it working in future versions.
- Does this solution deliver significant enough added value to make it worth doing?
- Value 1: Access to central permit/deny system with GUMS
- Value 2: Use similar automated model for VO tagging between grid and cloud
- Value 3: Using same code (privilege.jar) to access GUMS from cloud
- Value 4: Well-understood model of VO trust to accept other than KCA certs.

Arguments against X.509 AuthZ

- Relying on user to pick an FQAN, can't get info reliably from voms proxy.
  - (but GUMS + VOMS Admin ensures that they can't pick an unauthorized one and can't change it).
- One of a kind code—we are only site using combination of cloud + GUMS.
-

Future Challenges: Authentication

- OpenNebula X.509-authenticated Query/ReST is one-of-a-kind
  - Used by other European cloud sites still?
  - Need to find out who and how.
  - Appeal of ReSTful interface to API developers because you don't have to deal with presenting X.509 certificates.
  - Some sites use host cert for TLS-based https:// access, there are instructions in ONe manual on how to do that.
- This work begins only after ONE 4.x is deployed with X.509 authentication.

What is OpenStack doing?

- Need to understand “Keystone” authentication model of OpenStack
- Several people tried VOMS addons to OpenStack, did any of them succeed?
- CERN authenticates against LDAP server only at account creation, normal access/secret key after that.

Decision that has to be made:

- Amount of work to keep X.509 authentication going compared to:
  - Effort to switch to something else
  - Security risks we would take on if we did.
  -

What is future of AWS?

- They say they will drop X.509 authenticated SOAP API.
- OpenNebula never had EC2 SOAP API; we believe that OpenStack never had EC2 SOAP API.
- What will take its place?
  - Https: TLS-based access and secret key on ReST interface?
  - Something else entirely?
  - Mention of new protocol in One release notes?
- Need to understand what AWS is doing.
- Can HTCondor play with ReST API @AWS?