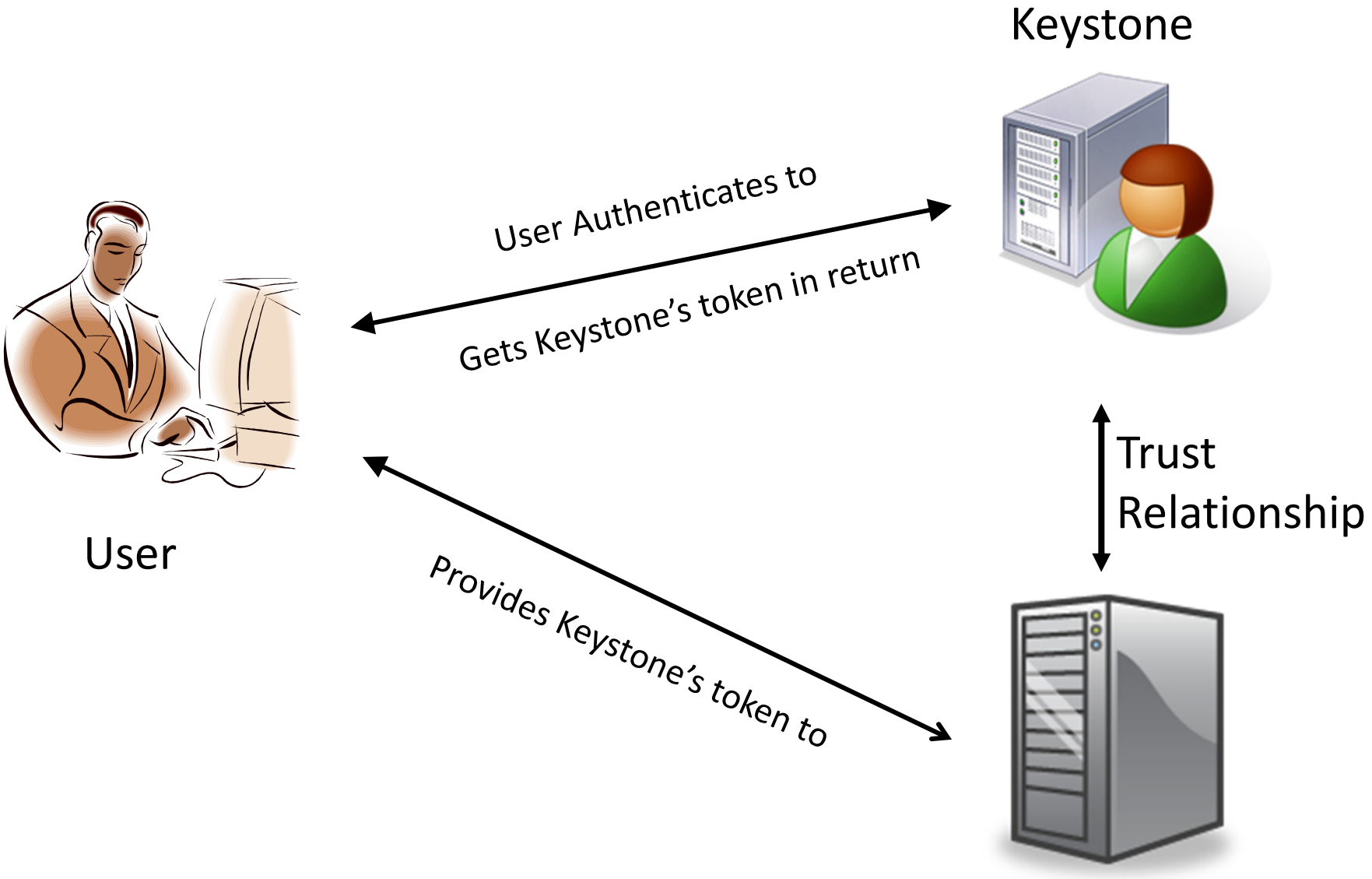
Adding FIM to Openstack

David Chadwick
University of Kent

Contents

- How OpenStack works
- Our first FIM implementation
- Our second FIM implementation
- The official OpenStack release (scheduled for April 2014) – still tentative

Authentication in OpenStack

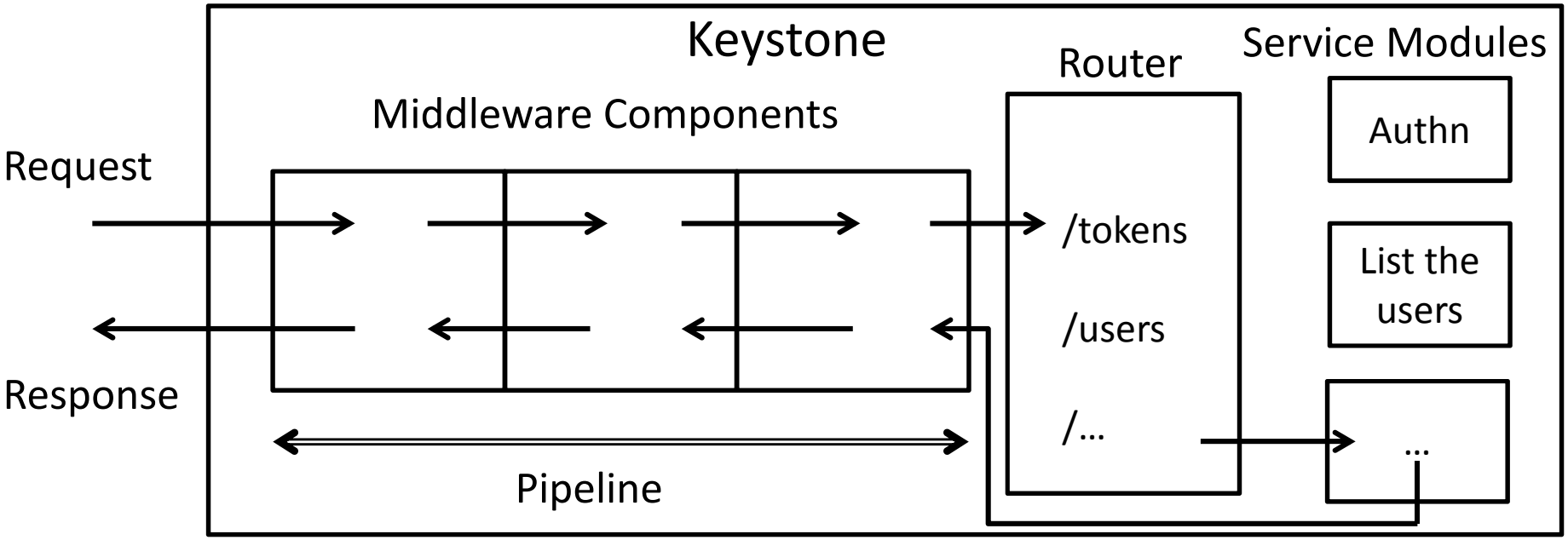


Authorisation in OpenStack

- Keystone token contains user's ID and roles
- Services then use either user's roles and RBAC to grant access to resources, or user's ID and DAC

- In order to add FIM to OpenStack we do not need to change any of the OpenStack services provided Keystone still returns the same token as in the non-federated case
  - Services will be ignorant of federation

Keystone Internal Architecture



Keystone Authn Module

- Keystone's authentication module supports multiple authn methods, each as plugins.
- Password and External are provided as core components. Users can also define their own
- Password uses backend LDAP to authenticate user
- External is for when Keystone is run in Apache HTTP Server (using mod_wsgi) and it passes the authenticated username to Keystone using the REMOTE_USER environment variable

Kent's Initial Implementation - Protocol Independent Pipeline Plugin

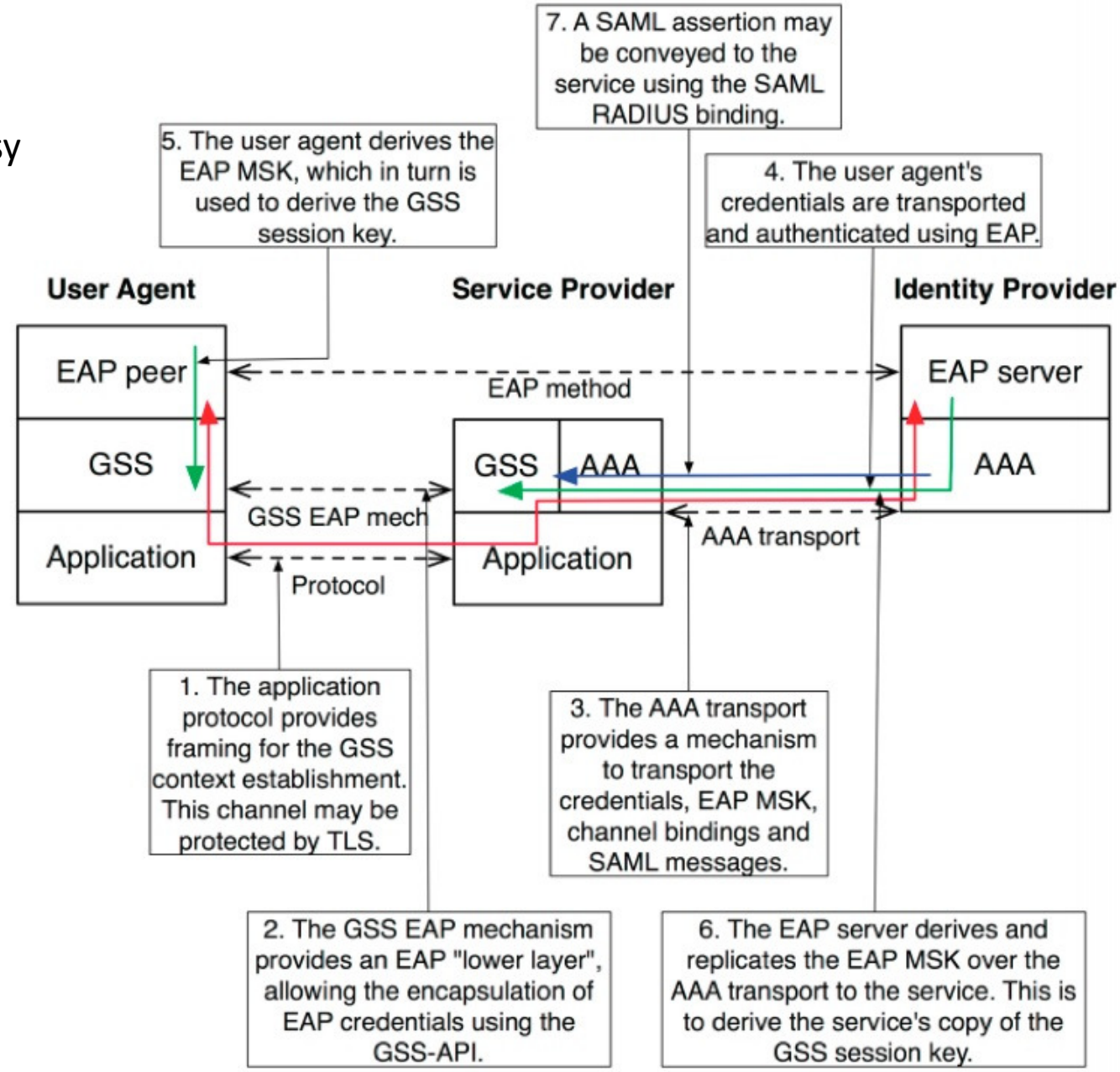
- Chosen because easiest for admins to add FIM – only need to change Keystone config file. No code changes needed
- FIM Plugin has three protocol dependent methods
  - Get IdP Request – get protocol specific request message to be sent to IdP
  - [Negotiate Parameters – optional for those protocols that need it such as ABFAB]
  - Validate IdP Response – protocol specific way of validating IdP's response
- Common output at the end

FIM Protocol Output

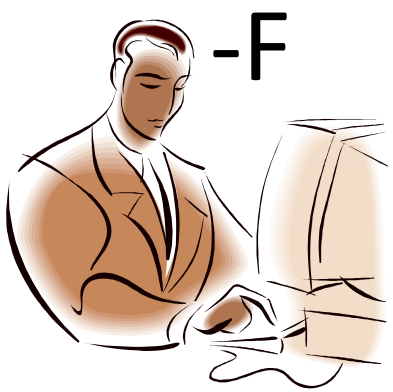
- Federation wide Unique ID of end user
- Set of {Set of user identity attributes and name of IdP that asserted them}
  - Caters for future attribute aggregation
- Validity time of asserted identity

ABFAB – SAML EAP Profile

Picture courtesy of BeSTGRID University of Auckland, NZ



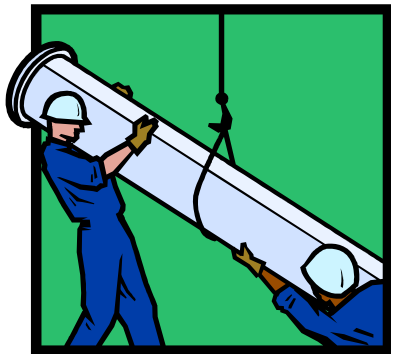
Federated Authentication



Modified Client Software

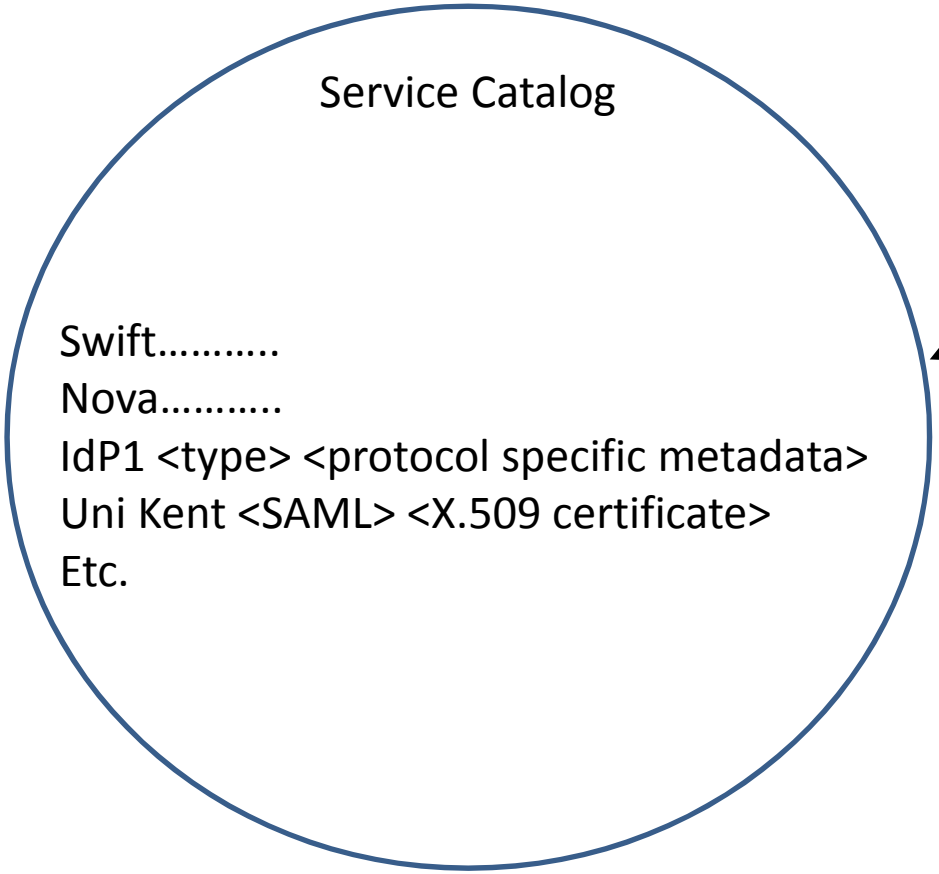
X-Authentication-Type: federated

Protocol Independent
Federation Handling



Keystone Pipeline

Trust in IdPs



Keystone Admin

If an IdP is not in the Service Catalog it cannot be seen or used by the user

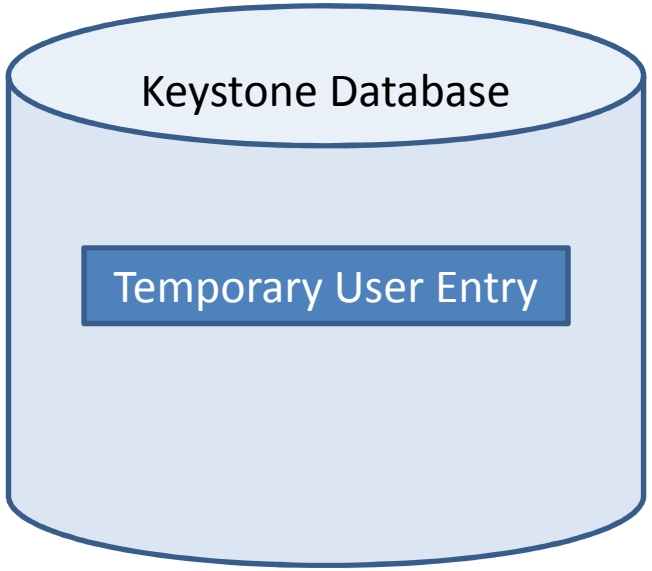
Trust in IdP's Attributes

- A table stores list of attributes (types and optional values) that each IdP is trusted to issue
- If asserted attributes are not in this table, they are thrown away by the protocol independent code

Gory Details

- X-Authentication-Type: federated header only
  - Performs Discovery. Returns list of IdPs from Service Catalog to client
- Header plus Body contains a JSON array with the chosen IdP in “idpRequest” element
  - Call protocol specific module ‘Get IdP Request’ method and return to client
- Header plus Body contains JSON array with “idpNegotiation” element
  - Call protocol specific module ‘Negotiate Parameters’ method and return to client
- Header plus Body contains a JSON array with an "idpResponse" element
  - Call protocol specific module ‘Validate IdP Response’ method

Magic 1 Auto Provisioning



Magic 2 - Attribute Mapping

Converts it into

IdP asserted identity (set of trusted identity attributes)

OpenStack recognised identity (roles, projects, domains)

Summary of Key Features

- Modular Design
- Most functionality is provided by protocol independent code we have added to Keystone's pipeline
  - Adding/Retrieving IdPs to *enhanced Service Catalog*
  - *Attribute Issuing Policy* creation and enforcement - says which IdPs are trusted to issue which identity attributes to users
  - Creating and removing *temporary user entries* in Keystone
  - *Attribute Mapper* from IdP issued identity attributes into Keystone roles, projects and domains
  - *Delegating* permissions to IdP administrators to set up the attribute mappings and attribute issuing policies
- One plug-in module needed that handles the *Protocol Specific* features of federated login
  - IdP Request preparation
  - idP Protocol negotiation (optional)
  - IdP Response verification
- Obviously clients have to be tailored to support federated login

Second Implementation – A new Federated Authn Method

- Took first implementation to Keystone developers for comment.
- They suggested we create a new Authn method, which they would integrate into a future release
  - So we moved the pipeline code to be a new Authn method called Federated
  - Produces a cleaner implementation. Does not need X-Fed header
- They said mods they were currently working on would not require us to keep creating temporary Keystone entries, as tokens could be issued for external users not in Keystone's database
  - So we removed this code

Federated Authn Module Validation

- Four working implementations:
- SAML plugin based on pySAML – now an operational service in Brazilian academic network
- Keystone plugin – for federating multiple OpenStack/ Keystone installations together
- ABFAB plugin based on Moonshot software
- OpenID Connect plugin (written by PhD student in Brazil)

Planned OpenStack April Release

- Keystone core developers decided to do a first quick fix for SAML only using Apache and mod_shib, and modifying the External authn method to pick up Remote_User and user's attributes as environmental parameters
- Will use the attribute mapping functionality from Kent's design/implementation to obtain the OpenStack roles and domains
- This week the core Keystone coders are meeting in Texas for a "hackathon" to get something working in time for the April release (codenamed Ice House)

What's Next?

- We no longer need the Federated Authn protocol independent module if the trust management code is moved up to the Authn level to cater for all Authn methods including External
- Thus our protocol dependent modules can become Authn methods in their own right
- We have just written a SAML ECP module for command line clients that can't use Apache
- Next we need to work on support for VOs and Communities of Interest from ABFAB