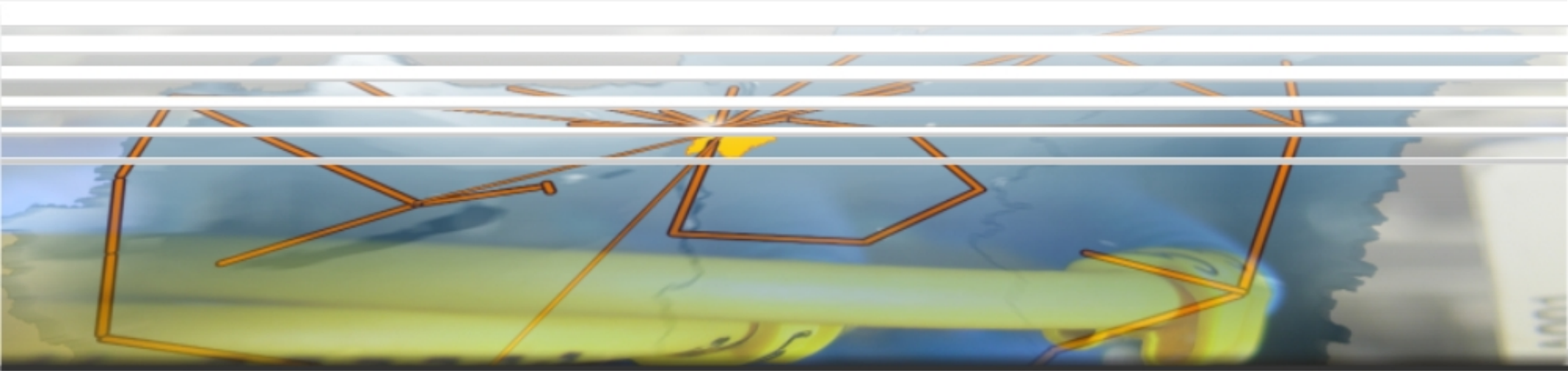


Challenges of (Really) Federating Applications



EuroCAMP 2009
Budapest
17th November 2009

Kristóf Bajnok
NIIF Institute



Preface

- Based on true story...
 - lessons we have learnt from integrating Drupal CMS to Shibboleth
 - might be applicable for
 - other apps
 - SimpleSAMLphp
- Checklist for application integration

- Interface with the SP implementation
 - Shibboleth
 - environment or headers
 - you have to write functions that process them
 - hint: Apache `mod_rewrite` ruins your environment, you have to use `ShibUseHeaders` directive
 - SimpleSAMLphp
 - API
 - `SimpleSAML_Auth_Simple()`
 - `isAuthenticated()`, `requireAuth()`, `logout()`
 - `getAttributes()`, ...

The Simple Way

- **SSO-enabling** applications
 - suitable for many use-cases
 - mostly intra-campus
- Presumes
 - non-opaque, persistent 'federated' identifier
 - such as eduPersonPrincipalName
 - application is organization-centric
 - user = organizational identity
 - privacy is handled elsewhere

Simple Prototype

```
/* Example PHP-like Shibboleth integration */
if (isset($_SERVER['Shib-Identity-Provider'])) { // we have SP session
    $uname = $_SERVER['REMOTE_USER']; // XXX what if missing?
    $user = DB::load_user($uname); // load user object from DB
    if (!$user) {
        // new user registration
        $user = new User("$uname"); // XXX illegal chars? (such as [:/!@])
        $user->setPassword(gen_random_pass());
        DB::save_user($user); //XXX numeric id? error handling?
    }
    echo "<p>Hello" . $user->uid . "</p>";
} else {
    // no SP session
    display_login_link(); // This provides link to a SessionInitiator
}
```

Getting deeper into it...

- Authorisation
- Understanding sessions
 - and also Logout
- Creating app-specific user entries
 - optional attributes
 - federated user identifiers
 - account linking
 - deprovisioning

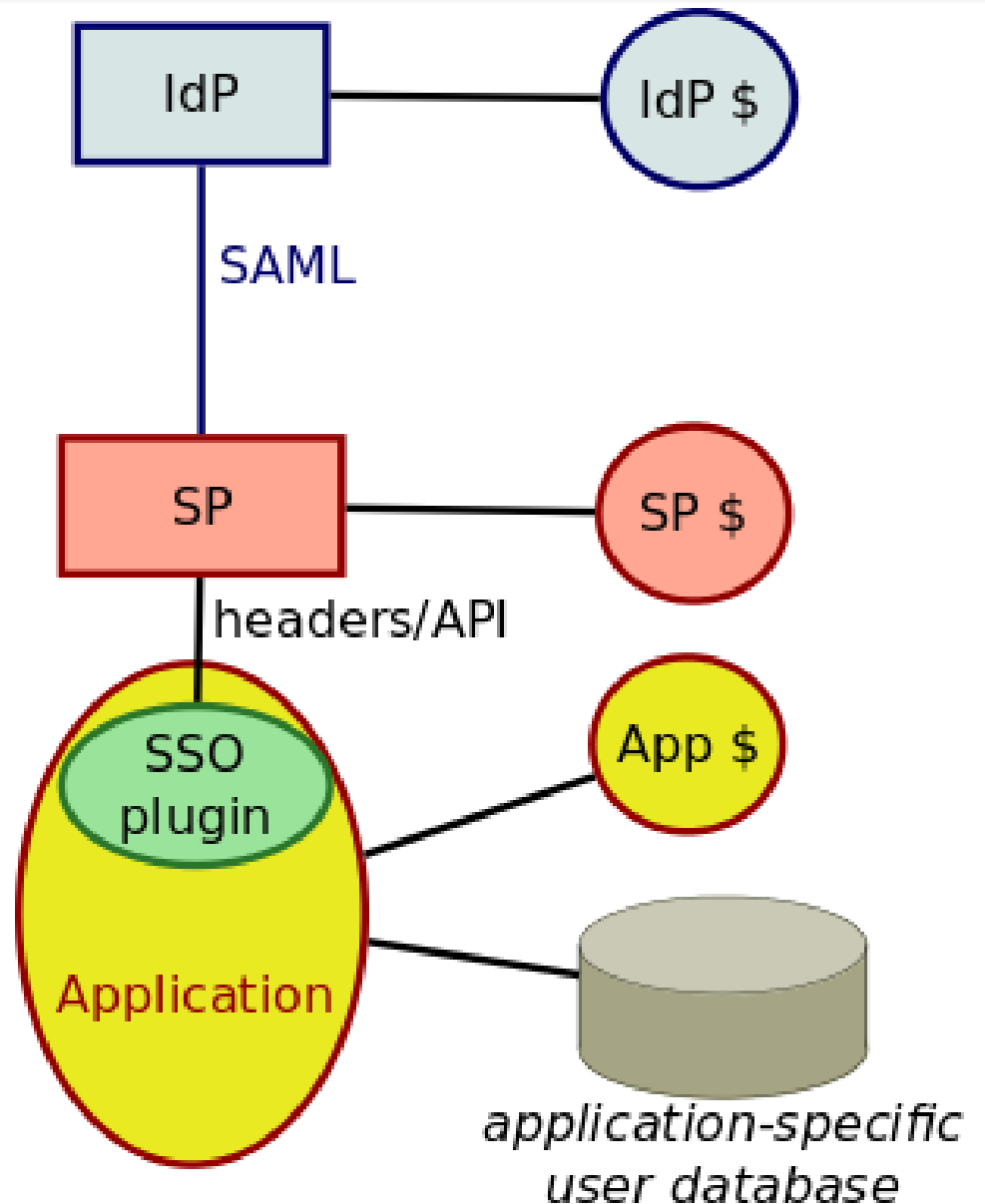
Authorisation

- Very application-specific
- Map attributes to roles or groups
 - eduPerson(Scoped)Affiliation
 - eduPersonEntitlement
- Entitlements can be grouped and managed through a VO platform

- **Static or dynamic?**
 - **Static permissions: saved along with user's database entry**
 - revocation is harder
 - user object caching issues
 - **Dynamic permissions: evaluated run-time**
 - impossible to get a consistent overview of who possesses which rights
 - only if managed outside of the application
 - **Hint: stay static if there is only one authorising module of the application**

Sessions and caches

- IdP session
 - caches authentication
- SP session
 - caches attributes received from IdP
- Application session
 - stores operational data
 - often caches user object
 - should be bound to SP session



Binding SP session & application session together

- Goals

- terminate SP (and optionally) IdP session on application logout
- terminate application session when the SP session ends
- notice if the SP session belongs to somebody else

Application logout → Shibboleth logout

- Shibboleth logout
 - Local logout: SP-only
 - little use if there is an IdP session
 - Global logout (Single Logout, SLO)
 - SP and IdP *and other SPs and other applications*
 - **google Shibboleth IdP Single Logout!**
- Application logout first!
 - remove application session
 - redirect to the Shibboleth LogoutInitiator and specify a reasonable a return URL
 - note: default Shib SP tries SLO but default Shib IdP does not support it → error message
- Otherwise it's not guaranteed that the application session is removed

Shibboleth logout → Application logout

- The easy way:
 - on every page access, verify that the SP session is still valid, make the user anonymous if it's not
 - the application gets notified on the next page refresh (if it's ever occur)
- Drawbacks
 - it's not possible to do real-time logout
 - it's not possible to display a reliable list of users logged-in

Shibboleth logout → Application logout

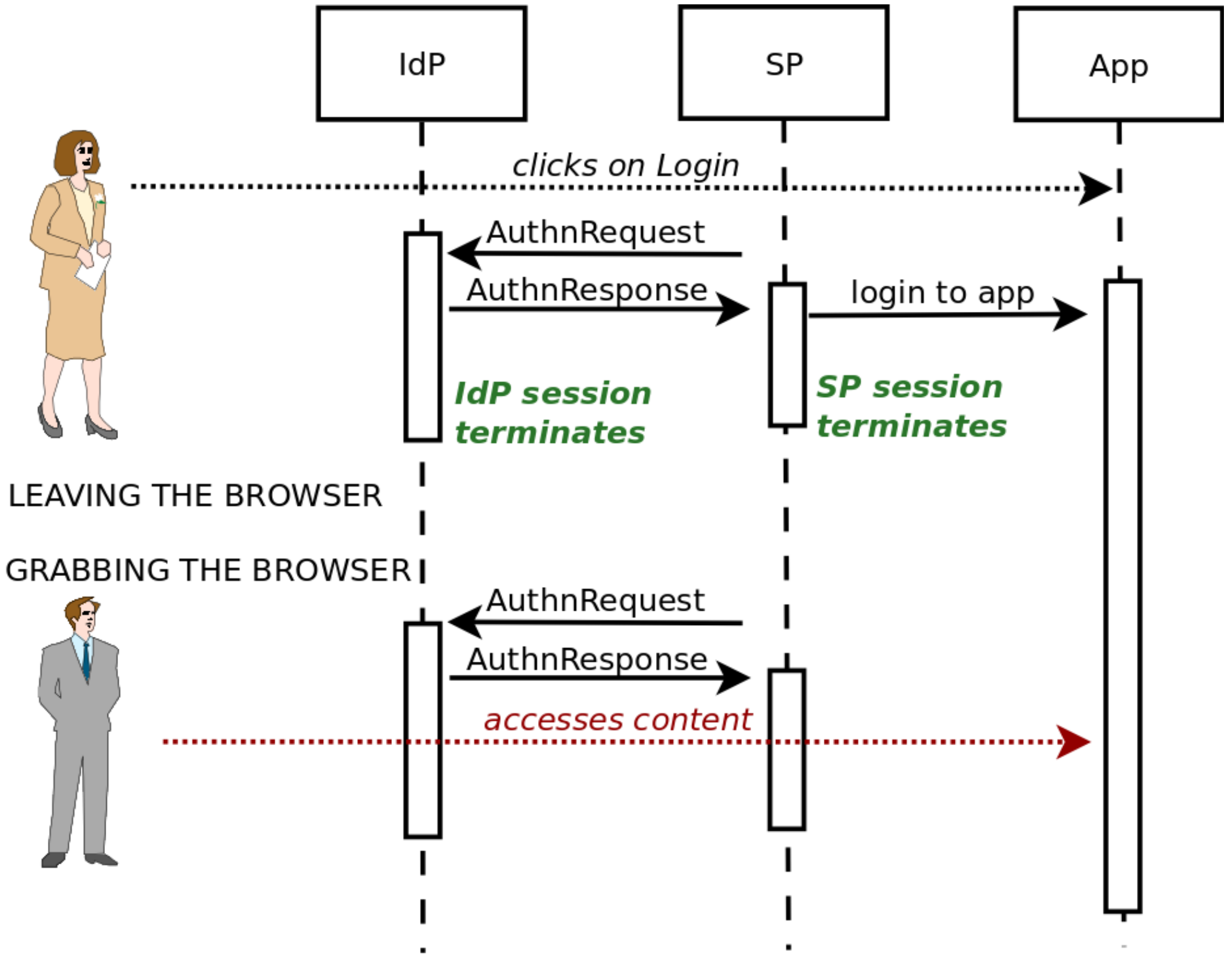
- The Hard Way (Shib-only): **SP Notify**
 - SP can notify the application on certain events:
 - Logout, [NameID Management]
 - but not on normal session expiry
 - via front-channel or back-channel webservice call
 - administrative logout will be possible only with back-channel logout, for which back-channel notification is necessary
 - application needs to implement the web service
 - front-channel: remove session and redirect
 - easier because you have the cookies
 - back-channel: SOAP server
 - retrieving sessions requires more advanced application session handling (eg. DB) and an auxiliary index on the SP session ID

<https://spaces.internet2.edu/display/SHIB2/SLOWWebappAdaptation>
<https://spaces.internet2.edu/display/SHIB2/NativeSPNotify>



SP session changes

- Check for SP session changes!
 - and learn from our mistake:
DRUPAL-SA-CONTRIB-2009-070



Attack description

- Application session has a long lifetime
- Alice's IdP and SP session are removed without the application knowing
 - Single Logout or normal timeout
- Using the same browser, Bob creates a session at the SP with his own IdP
 - but not through the first application
 - unsolicited session initiation is allowed with Shib
- Bob can access the application using Alice's session
 - and if user info is cached in the application session... :(

Notes on the attack

- Using SP notification does not save you
 - no notification on normal expiration
 - but will help with the SLO-cases
- Generally you should bind your application's session to the SP session
 - expiry
 - indexing
- Workaround if it's too much effort:
 - **store the persistent federated identifier in the user's application session**
 - and reset the session if a change is found

Back to the agenda

- Authorisation
- Understanding sessions
 - and also Logout
- ***Creating app-specific user entries***
 - ***federated user identifiers***
 - ***account linking***
 - ***deprovisioning***

Application-specific user DB

- Persistent identifier

- as a key to the DB
- SHOULD be opaque, targeted
 - eg. eduPersonTargetedID

- this case it cannot be used as a username

<https://papigw.aai.niif.hu/idp/shibboleth!https://papigw.aai.niif.hu/shibboleth!oM0oNyt0B/YwRMIvEXMT4jAuBew=>

- Identifier mapping

- federated id → username
 - given by the user on first logon
 - optionally pre-filled from eduPersonNickName
- length
 - ask your federation operator to limit entityIDs
 - 255 bytes SHOULD be enough for the whole ePTID
 - you can leave out the SP part, but do not truncate blindly!

Account linking

- Multiple federated IDs can be mapped to one local username
 - basic **account linking**
 - only **authenticated users** may do that
 - technically: it's possible to replace the SP session with authenticating at another IdP
- Use cases:
 - where SSO is just auxiliary besides U/P
 - students attending to more than 1 inst.
 - academic users' lifecycle
- Roles and permissions can be different
 - if they are based on attributes

User consent

- If you store personal data in your app DB, you are a **data controller**
 - unless <ertain_legal_stuff_applies> the SP has to **ask for consent** before storing **any user-related data**
 - this differs from IdP consent tools like uApprove
- Multiple approaches possible
 - shib_auth:
 - display URL to the privacy policy + checkbox
 - privacy policy can be versioned
 - if the admin tells that the version has changed, the user needs to re-accept
 - no 'get your filthy hands off my data' button. Yet...

All in all: registration form on first login!

- Registration form can be cumbersome to develop, OTOH if you use it
 - you can ask for missing / optional attributes
 - loose your federated attribute requirements
 - may be pre-filled with values from the federation
 - let the user choose an identifier
 - you can ask for user consent

Deprovisioning

- Get rid of stale data in app DB
 - you can use `resolvertest` utility to do SAML2 attribute queries, provided you have
 - the federated identifier
 - and you know the Format
 - the IdP entityID
 - the IdP supports back-channel AttributeQuery
 - run it as a cron job on the SP
 - you can delete users based on changes in the attributes returned
 - or if the user is not found at the IdP
 - deleting users from an application can be really hard
 - history, id reassignment, ...
 - still preferred from privacy POV

Checklist

- Sessions
 - mind your app session lifetime!
 - prepare for back-channel notifications if possible
 - store your session in a DB
 - SP session ID as an index
 - verify the user ID stored in SP session
- Implement an app registration form for
 - human-readable usernames
 - optional attributes
 - asking consent on privacy policy
- Use identifier mapping
 - and account linking if appropriate
- Search for stale users