
entr'
Ouvert

Lasso

Implémentation libre de Liberty Alliance

Frédéric Péters
<fpeters@entrouvert.com>

Vandœuvre

- Projet « carte de vie quotidienne » de l'ADAE
  - Carte démocr@tics
- Standards PKCS11/15, X.509, etc.
- Respect de la vie privée
  - Qu'y stocker ?
  - rien ou un minimum

Mauvais exemple belge

- Certificat X.509
  - Nom, prénom, adresse, etc.
  - Numéro d'identification au registre national
  - Consigne « il vous est communiqué, vous ne pouvez pas le stocker »

Custom Identity Mapping

Capture certificate information on server

```
Private Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    Handles MyBase.Load

  Dim clientCertificate As HttpClientCertificate = Request.ClientCertificate

  With clientCertificate
    Label1.Text = .Issuer
    Label2.Text = .SerialNumber
    Label3.Text = .Subject
    Label4.Text = .ValidFrom.ToString()
    Label5.Text = .ValidUntil.ToString()
    Label6.Text = .ServerIssuer
    Label7.Text = .IsPresent.ToString()
    Label8.Text = .ServerSubject
    Label9.Text = .IsValid.ToString()
  End With
End Sub
```

Don't forget RRN can't be stored by non-governmental orgs

Pas assez

- Consultation de la population
  - Besoin d'anonymat
  - Éviter l'identifiant unique qui permettrait corrélation
- Solution propre ?
- Existant ?

Liberty Alliance

- Consortium fondé mi 2001 par Sun
- Plus de 160 membres
  - Sun, IBM, Nokia, France Telecom, etc.
- Alternative au projet Passport de Microsoft standard ouvert.
  - Une norme, pas un site
- Validé par l'Union Européenne sur l'aspect « vie privée »
  - « Working document on online authentication services » adopté le 29 janvier 2003

Obstacles à Liberty

- Pas de solutions libres
  - Implémentation de référence par Sun mais uniquement 1.0
  - SourceID, accès au source mais pas libre
    - Maintenant, libre, licence semblable à la GPL
- Solutions existantes
  - Intégration difficile, très peu de flexibilité

Développement de Lasso

- Prototype en Python
  - (juillet 2003)
- Réimplémentation en C
  - (février 2004)

Les acteurs

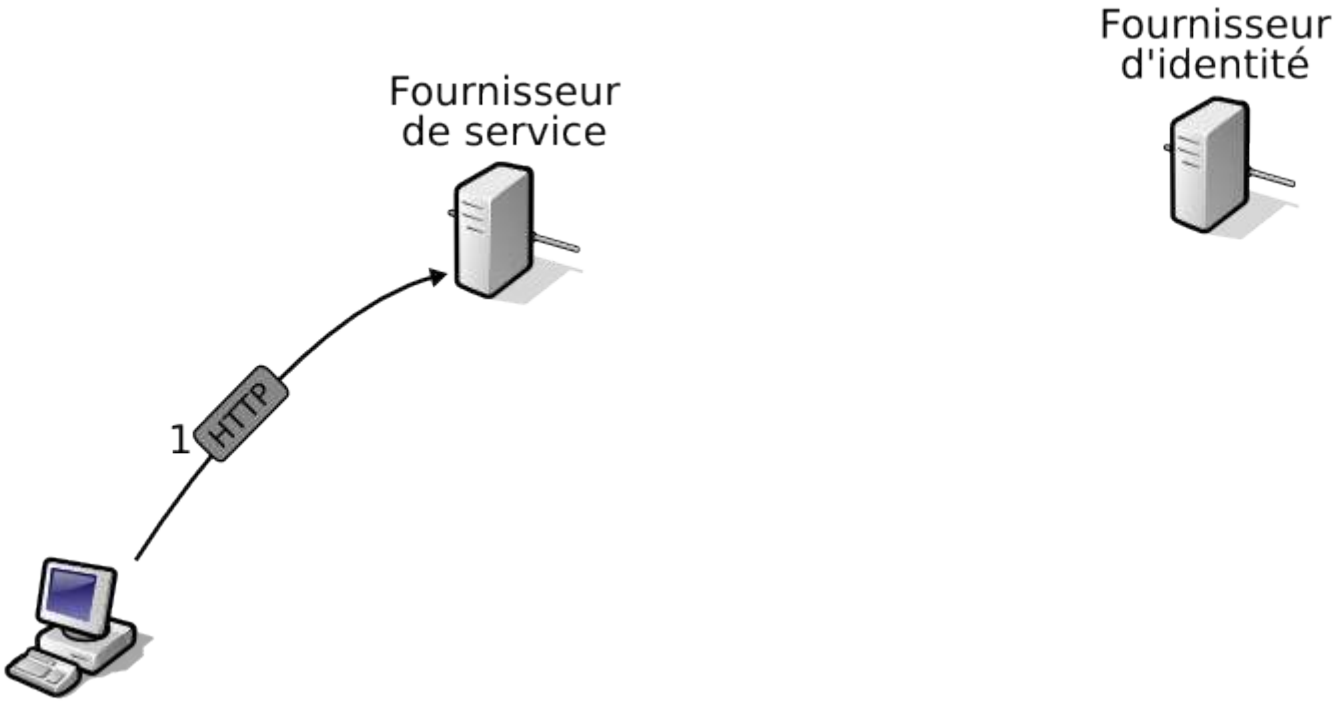
- 3 types d'acteurs :
  - l'**utilisateur** : personne physique ou morale qui peut acquérir une identité ;
  - le **fournisseur d'identité** (IdP) : crée, et gère l'identité des utilisateurs, et les authentifie auprès des fournisseurs de service ;
  - le **fournisseur de services** (SP) : fournit des services aux utilisateurs une fois qu'ils sont authentifiés par un fournisseur d'identité.

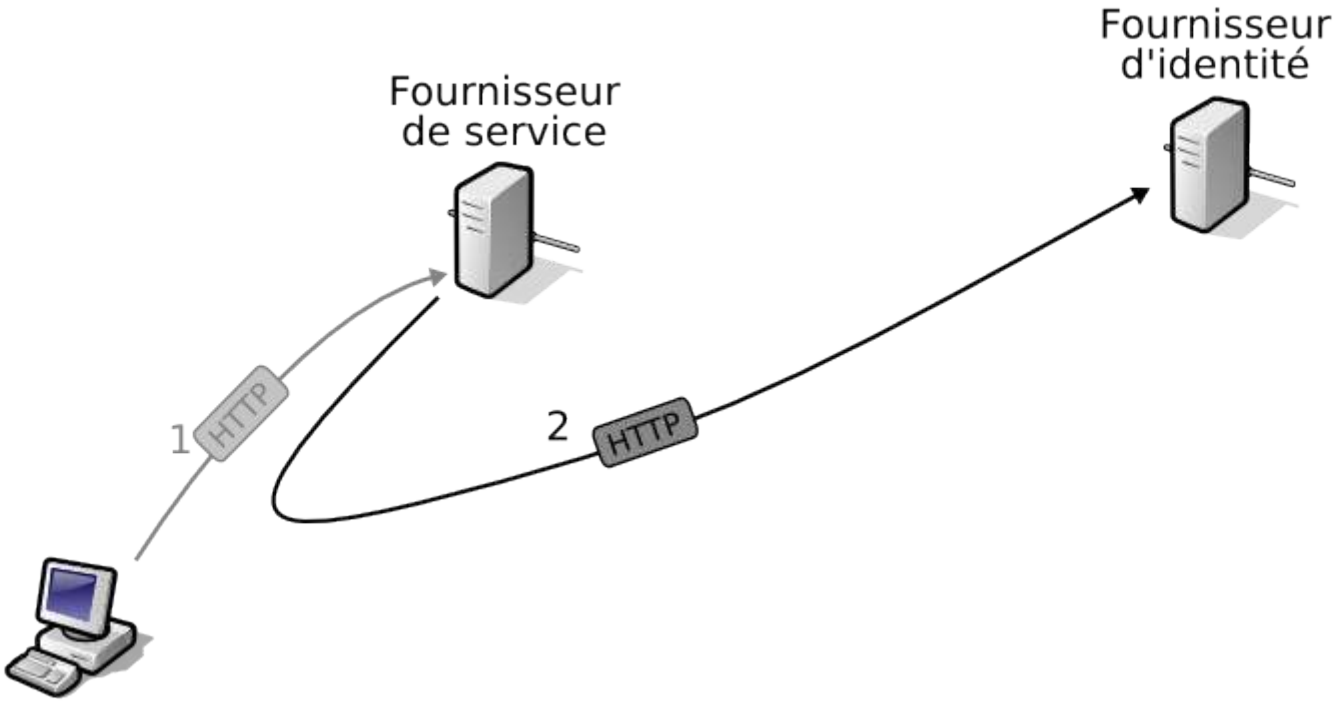
Single Sign On

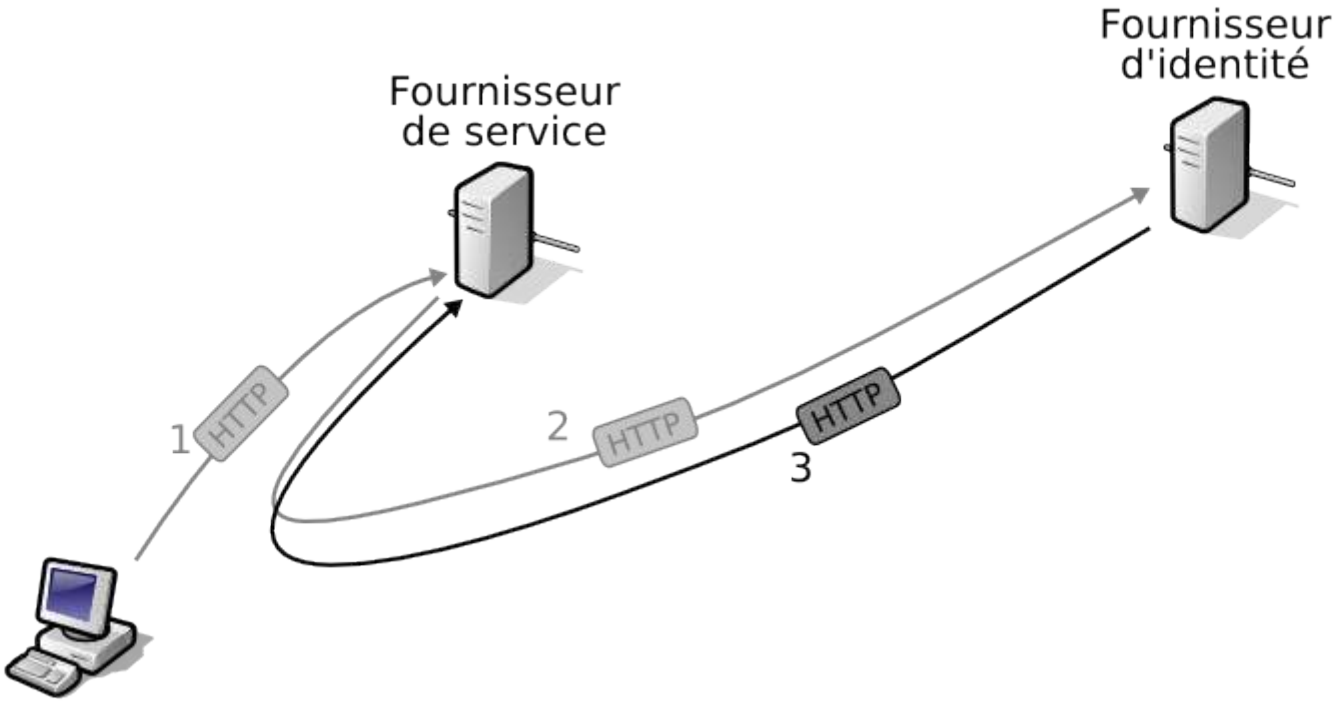
- L'identification unique (Single Sign On ou SSO) permet à un utilisateur de s'identifier auprès d'un IDP, et d'être ensuite automatiquement identifié sur tous les SP appartenant à son cercle de confiance.
- Les différents SP ne peuvent communiquer directement entre eux à propos d'un utilisateur.

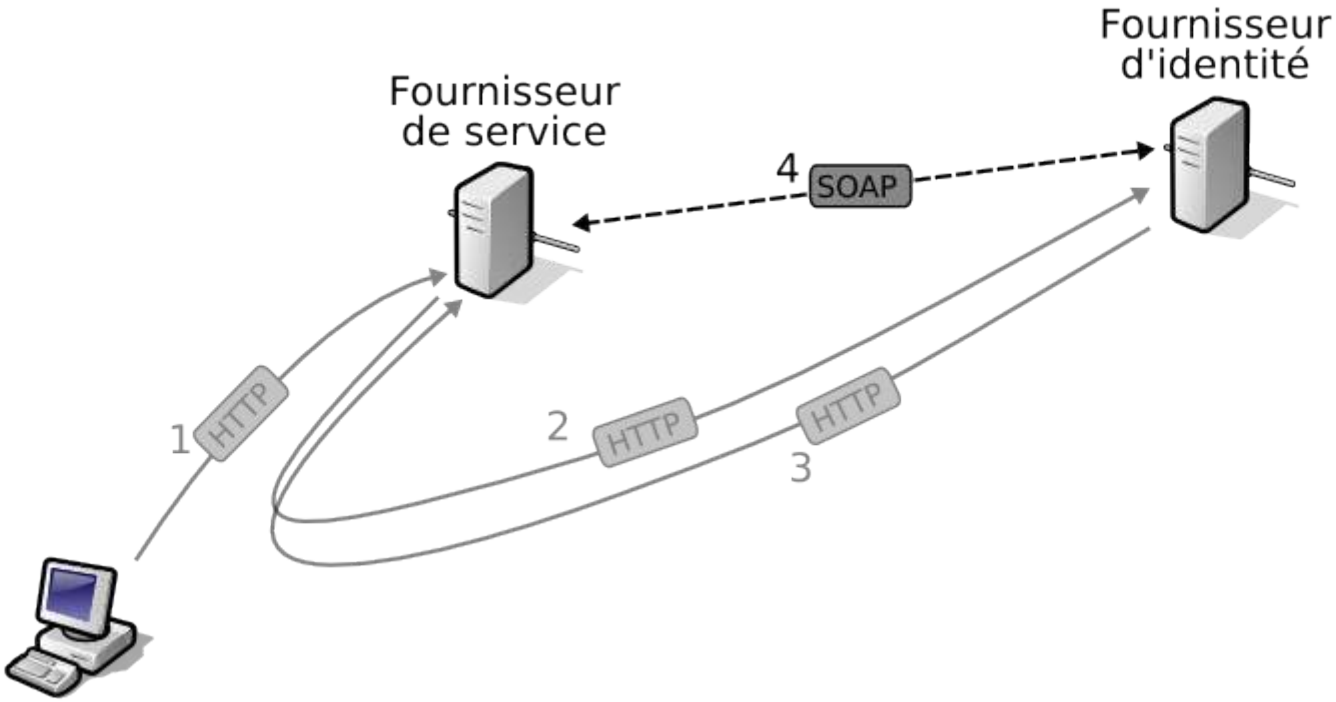
Le Single Sign On Liberty

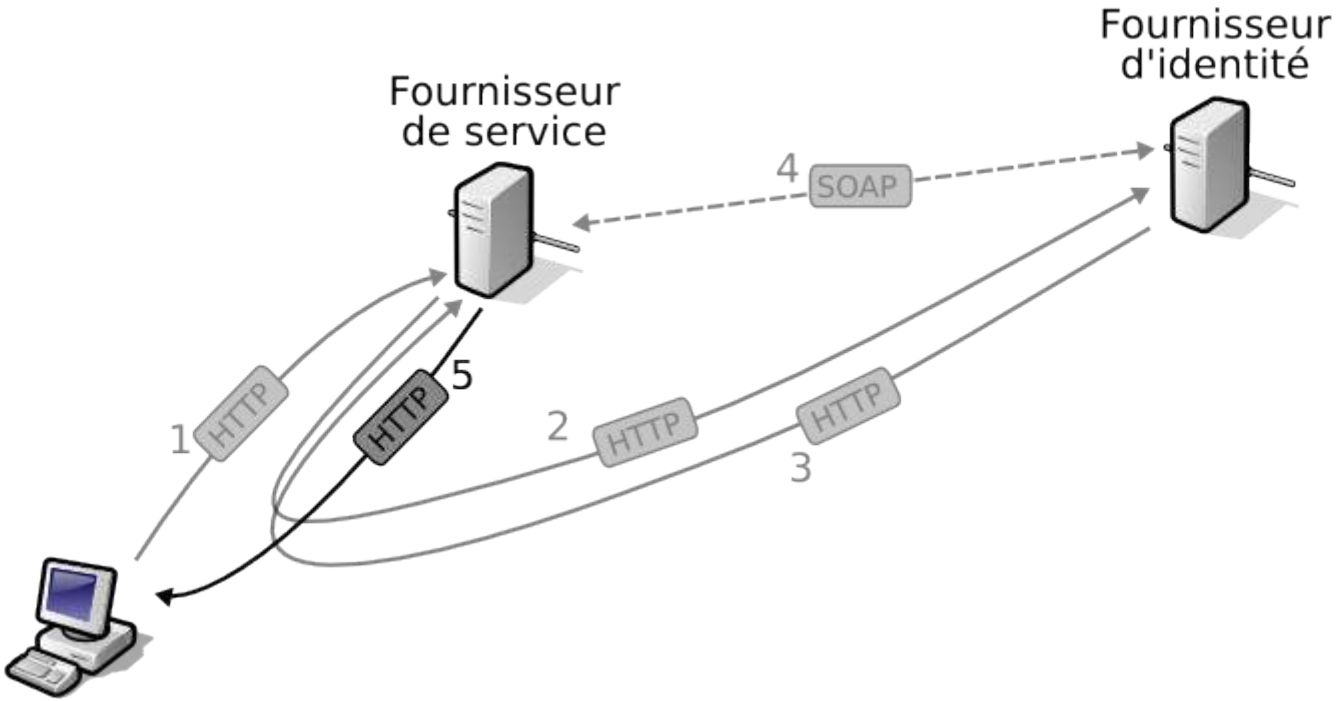
- 260 pages de specs (au moins)
  - mais aussi Single Logout, Défédération, etc.
  - basée sur SAML (standard de l'OASIS)
  - http://www.projectliberty.org/specs/
- Différents « profils »
  - *Artifact* : 5 flèches


Fournisseur
de service
Fournisseur
d'identité
1
HTTP


Fournisseur
d'identité
Fournisseur
de service
1
HTTP
2
HTTP


Fournisseur
d'identité
Fournisseur
de service
1
HTTP
2
HTTP
HTTP
3


Fournisseur
d'identité
Fournisseur
de service
4
SOAP
1
HTTP
2
HTTP
HTTP
3


Fournisseur
d'identité
Fournisseur
de service
4
SOAP
1
HTTP
HTTP
5
2
HTTP
HTTP
3

Lasso

Caractéristiques fonctionnelles

- Bibliothèque
  - Algorithmes spécifiés par Liberty
- Pas de couche stockage
  - Pas d'obligation de base de données, de schéma de tables particuliers, etc.
- Pas de couche transport
  - Utilisation de celles de l'applicatif
- Pas de couche authentification
  - Du ressort de l'applicatif

Caractéristiques techniques

- Bibliothèque écrite en C
  - Rapide
  - Compatible avec un maximum de langages
    - Java, Perl, PHP, Python tout à fait fonctionnels, C# en route
- Utilisation de SWIG pour ces bindings
  - Seulement quelques jours pour porter vers un nouveau langage
- Multi-plateforme
  - Testée sous GNU/Linux, FreeBSD, Mac OS X et Microsoft Windows

Bibliothèques

- S'appuie sur des bibliothèques courantes, écrites en C et libres :
  - GLib et Gobject: portabilité, listes, dictionnaires, structures *objet* en C
    - http://www.gtk.org
  - libxml2: traitement XML
    - http://www.xmlsoft.org
  - XMLSec: XML Signature, XML Encryption
    - http://www.aleksey.com/xmlsec/
  - OpenSSL: crypto
    - http://www.openssl.org

XMLSec, recommandation W3C

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
<CanonicalizationMethod Algorithm="..."/>
<SignatureMethod Algorithm="..."/>
<Reference URI="#_2816FDE05EB330F05FEAC8E98D8A3E30">
<Transforms>
<Transform Algorithm="..."/>
<Transform Algorithm="..."/>
</Transforms>
<DigestMethod Algorithm="..."/>
<DigestValue>+5jOoBkeDMh6xTuMmJtsedleKHs=</DigestValue>
</Reference>
</SignedInfo>
<SignatureValue>JgbQRybCsH1/734DSFKsdfsdxcwxc...</SignatureValue>
<KeyInfo>
<X509Data>
<X509Certificate>MIIEKzCC23sdAZE2Sqwdsdsdszr1...</X509Certificate>
</X509Data>
</KeyInfo>
</Signature>
```

Interopérabilité

- Testée avec SourceID Liberty 2.0 beta demo application
  - http://www.sourceid.org
  - http://lasso.entrouvert.org/interoperability
- Prochainement:
  - Événements « *conformance* » de Liberty Alliance
  - Avril ou juin

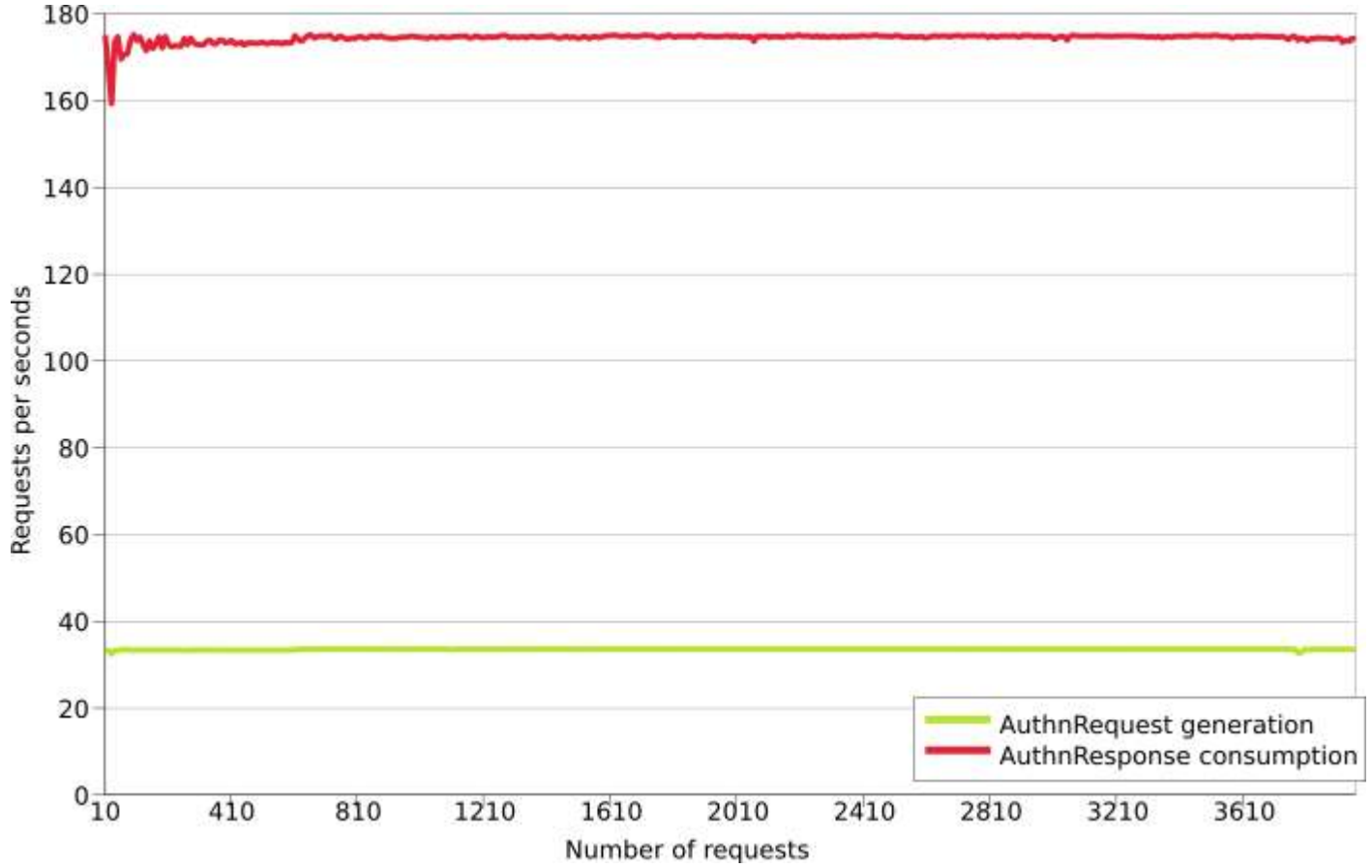
Licence

- Licence GNU GPL
- Copyright exclusif Entr'ouvert
- Donc:
  - Utilisation possible dans les applications ayant une licence compatible avec la GPL
  - Utilisation possible dès lors que l'application n'est pas distribuée
  - Autre cas ? licence payante, nous consulter

Performances (1)

- 92% du temps passé dans OpenSSL
- Énormes gains possibles avec processeurs dédiés
  - mais un *petit* Opteron (1,6GHz) assure déjà 140 requêtes/secondes

Performances (2)



Contrôle qualité

- http://lasso.entrouvert.org/buildbox

Build time	Changes			Compilation		Components					Tests		
	Nb	Log	Guilty*	Duration	Build log	Lib C	Python	PHP	Java	C#	Lib C	Python	Souk
11/09 18:35	1	C	rchantereau	3:59	3.3 / 2.95								
16:35	1	C	cnowicki	3:53	3.3 / 2.95								
15:35	1	C	nclapies	3:56	3.3 / 2.95								
15:05	1	C	cnowicki	4:11	3.3 / 2.95								
12:05	1	C	fpeters	3:59	3.3 / 2.95								
11:35	0	C		4:00	3.3 / 2.95								
11:20	1	C	fpeters	3:45	3.3 / 2.95	L	L	L	L	L	L	L	L
10:35	0	C		4:06	3.3 / 2.95								
10:20	1	C	fpeters	4:10	3.3 / 2.95	L	L	L	L	L	L	L	L
11/08 20:05	1	C	fpeters	3:54	3.3 / 2.95								
19:20	2	C	fpeters	3:58	3.3 / 2.95			L					
18:50	3	C	fpeters	3:59	3.3 / 2.95			L					
18:35	1	C	nclapies	3:57	3.3 / 2.95			L					
18:05	2	C	cnowicki	3:59	3.3 / 2.95			L					
17:20	1	C	eraviart	3:55	3.3 / 2.95			L					
17:05	0	C		4:02	3.3 / 2.95			L					L
13:35	0	C		4:03	3.3 / 2.95			L					L
13:20	3	C	fpeters	3:52	3.3 / 2.95	L	L	L					L
11:50	1	C	nclapies	4:04	3.3 / 2.95			L					L

Applications

- IdPC
  - Un IDP en CGI & C
  - PostgreSQL, authentification HTTP ou par certificat X.509
- SPC
  - Un SP en CGI & C
- Souk
  - implémentation de référence de Lasso

Applicatifs Démocr@tics

- Fournisseur d'identité :
  - http://identification.lesdemocratics.net
- Consultations de la population :
  - http://consultation.lesdemocratics.net
- Formulaires administratifs :
  - http://formulaires.lesdemocratics.net
- Téléfact : télépaiement des factures parascolaires (CGI en C)
- Site principal de la mairie (ColdFusion)

Intégration de Lasso dans un SP

- Initialisation du SP

```
/* C */
LassoServer *server;
server = lasso_server_new(metadata, private_key,
     secret_key, certificate);
lasso_server_add_provider(server,
     LASSO_PROVIDER_ROLE_IDP, idp_metadata,
     public_key, certificates_chain);

# Python
server = lasso.Server(metadata, private_key,
     secrserver.addProvideret_key, certificate)
server.addProvider(lasso.ProviderRoleIdp,
     idp_metadata, public_key, certifacates_chain)
```

Intégration de Lasso dans un SP

- Envoi d'une demande d'authentification

```
login = lasso.Login(server)
login.initAuthnRequest(provider_id,
     lasso.HTTP_METHOD_REDIRECT)
login.request.isPassive = False
login.request.nameIdPolicy = "federated"
login.request.consent = lasso.LIB_CONSENT_OBTAINED
login.buildAuthnRequestMsg()

# reply with redirect, example in CGI:
print 'Location:', login.msgUrl
```

Intégration de Lasso dans un SP

- Traitement de l'assertion

```
login = lasso.Login(server)
login.initRequest(query_string)
login.buildRequestMsg()
# send login.msgBody to login.msgUrl (SOAP)
login.processResponseMsg(response)
nameIdentifier = login.nameIdentifier.content
# retrieve user account and session from
# nameIdentifier
login.setIdentityFromDump(identity_dump)
login.setSessionFromDump(session_dump)
login.acceptSso()
# store login session and identity
```

Développement

- Public
- Liste :
  - lasso-devel@lists.labs.libre-entreprise.org
- Bug tracking :
  - http://labs.libre-entreprise.org/projects/lasso

Futur

- Serveur d'attributs (ID-WSF)
  - en cours de développement
- Compatibilité SAML 2.0
- Scalp
- Intégration dans un maximum d'applications libres
  - SPIP, WordPress, webcalendar, mailman, sympa, etc.
- Compatibilité Shibboleth (universités) ?

Questions / Réponses

http://lasso.entrouvert.org