django-mellon - Development #19396

pouvoir charger un fichier de fédération

12 octobre 2017 11:58 - Benjamin Dauvergne

Statut:	Nouveau	**Début:**	12 octobre 2017
Priorité:	Bas	**Echéance:**	
Assigné à:		**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	

Description

On peut s'inspire du code en C dans mod_auth_mellon:
https://github.com/UNINETT/mod_auth_mellon/blob/master/auth_mellon_handler.c#L264

L'API utile dans lasso est lasso_server_load_metadata(),
http://doc.entrouvert.org/lasso/stable/lasso-LassoServer.html#lasso-server-load-metadata

La fonction à toucher dans djang-mellon est mellon.utils.create_server() notamment la partie qui appelle addProviderFromBuffer(), on peut peut-être introduire une nouvelle méthode sur les classes Adapter nommée get_federations() qui renverrait de quoi charger une fédération (chemin du fichier, chemin de la chaîne de certificats de validation, etc..).

Au passage ça peut-être sympa de corriger #13881 et #11565, notamment si on gère l'utilisation d'URL pour désigner le fichier de métadonnées de la fédération (ce qui serait ubercool), mais je verrai plutôt une mise en cache via django.core.storage (i.e. dans le répertoire media/ du tenant).

Demandes liées:

Lié à django-mellon - Bug #21067: Le commit de chargement d'un fichier de féd...	**Fermé**	**09 janvier 2018**
Lié à django-mellon - Bug #24531: crash tests tox avec lasso 2.6.0-1~eob90+1	**Fermé**	**14 juin 2018**

Révisions associées

Révision 63993e36 - 09 janvier 2018 17:50 - Paul Marillonnet

support federation file loading (#19396)

Révision 6c528dd2 - 09 janvier 2018 21:43 - Benjamin Dauvergne

Revert "support federation file loading (#19396)"

This reverts commit 63993e360c89b22ea8e70fabf12d4f7e223cfb90.

Historique

#1 - 12 octobre 2017 12:01 - Benjamin Dauvergne

Dans l'idée je vois un nouveau setting:

```
"FEDERATIONS": [
   "/etc/renater/renater.xml",
   "https://services.renanter.federation/etc..",
   ("/etc/renanter/edugain.xml", "/etc/renanter/edugain.pem"),
]
```

Si la fédération est un chemin on le charge, si c'est une URL pareil mais on met en cache (faut normaliser l'URL pour en faire un nom de fichier, genre avec slugify(), et garde que les 250 premiers caractères[1]), si c'est un tuple pareil pour le premier élément, le deuxième élément est le certificat pour la validation (idem mettre en cache, etc..).

[1] https://serverfault.com/questions/9546/filename-length-limits-on-linux

#2 - 12 octobre 2017 15:03 - Benjamin Dauvergne

En lien avec le #19260 il faudra aussi voir de ne bloquer sur get_idp(entity_id) ne retourner rien (actuellement ça cherche dans la liste des IdP explicitement listés dans la configuration si on charge un fichier de configuration il va falloir ruser pour que ça marche); il y a un peu de boulot d'analyse donc.

#3 - 13 octobre 2017 11:02 - Paul Marillonnet

Si tu es d'accord je vais déjà essayer d'avoir quelque chose de fonctionne avec le premier des trois formats de métadonnées que tu proposes, à savoir un fichier local.

J'utilise lasso.Server.loadMetadata() qui a l'air d'être le wrapping de la fonction lasso_server_load_metadata() de l'API C dont tu parles dans la description du ticket.

Je voudrais déjà me débrouiller pour faire fonctionner quelque chose sous cette forme :

```
$ git diff
diff --git a/mellon/app_settings.py b/mellon/app_settings.py
index aeeab73..aa4fcd0 100644
--- a/mellon/app_settings.py
+++ b/mellon/app_settings.py
@@ -36,6 +36,7 @@ class AppSettings(object):
         'LOGIN_URL': 'mellon_login',
         'LOGOUT_URL': 'mellon_logout',
         'ARTIFACT_RESOLVE_TIMEOUT': 10.0,
+        'FEDERATIONS': [],
     }

     @property
diff --git a/mellon/utils.py b/mellon/utils.py
index ab092a7..1be395a 100644
--- a/mellon/utils.py
+++ b/mellon/utils.py
@@ -83,6 +83,14 @@ def create_server(request):
                 logger.error(u'bad metadata in idp %r', idp['ENTITY_ID'])
                 logger.debug(u'lasso error: %s', e)
                 continue
+        for federation in app_settings.FEDERATIONS:
+            try:
+                server.loadMetadata('LASSO_PROVIDER_ROLE_BOTH', federation, None, None, None,
+                                    'LASSO_SERVER_LOAD_METADATA_FLAG_DEFAULT')
+            except lasso.Error, e:
+                logger.error(u'bad metadata for federation %r', federation)
+                logger.debug(u'lasso error: %s', e)
+                continue
         cache[root] = server
         settings._MELLON_SERVER_CACHE = cache
     return settings._MELLON_SERVER_CACHE.get(root)
```

Quitte à tout mettre après dans une méthode adapters.DefaultAdapter.get_federations() comme tu le suggères.

Je vais aussi chercher une façon adéquate de tester au fur et à mesure la procédure de chargement du/des fichiers. Si tu as des pistes là-dessus, je suis preneur stp.

Est-ce que tu saurais stp me dire quels fichiers de métadonnées parmi ceux situés à l'adresse
https://metadata.federation.renater.fr/renater/main/
sont censés se charger correctement dans mellon ?

#4 - 13 octobre 2017 15:12 - Paul Marillonnet

Paul Marillonnet a écrit :

> Je vais aussi chercher une façon adéquate de tester au fur et à mesure la procédure de chargement du/des fichiers. Si tu as des pistes là-dessus, je suis preneur stp.

Pour ce faire, je suis en train d'écrire un fournisseur de service minimal déployé en local avec un authentic (lui aussi sur ma machine virtuelle locale). Je vais voir comment le chargement de fichiers de fédération s'agence dans mellon par rapport au SP et à authentic.

#5 - 13 octobre 2017 15:42 - Benjamin Dauvergne

Paul Marillonnet a écrit :

> Si tu es d'accord je vais déjà essayer d'avoir quelque chose de fonctionne avec le premier des trois formats de métadonnées que tu proposes, à savoir un fichier local.
>
> J'utilise lasso.Server.loadMetadata() qui a l'air d'être le wrapping de la fonction lasso_server_load_metadata() de l'API C dont tu parles dans la description du ticket.

Oui.

> Je voudrais déjà me débrouiller pour faire fonctionner quelque chose sous cette forme :
> [...]
> Quitte à tout mettre après dans une méthode adapters.DefaultAdapter.get_federations() comme tu le suggères.

Hmm, non faut charger ça là ou l'objet lasso.Server() est créé, tu n'as pas le choix.

> Je vais aussi chercher une façon adéquate de tester au fur et à mesure la procédure de chargement du/des fichiers. Si tu as des pistes là-dessus, je suis preneur stp.

Il faut que tu commences à savoir écrire des tests, avec tox/py.test/webtest et tout le tintouin, il y a des exemples déjà dans tests/.

Ok mais prend bien en compte le code dans LoginView.get_idp() qui n'est actuellement vraiment pas adapté à ce fonctionnement (il parcourt une liste explicite d'IdPs pour trouver le bon).

> Est-ce que tu saurais stp me dire quels fichiers de métadonnées parmi ceux situés à l'adresse
> https://metadata.federation.renater.fr/renater/main/
> sont censés se charger correctement dans mellon ?

Les idps suffisent, i.e. le fichier d'1,3Mo.

#6 - 16 octobre 2017 18:29 - Paul Marillonnet

- Patch proposed changé de Non à Oui

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

Ok merci pour tes conseils.

Pour info, j'ai commencé à écrire quelque chose pour le support du chargement d'un fichier local (cf patch WIP).
Je ne veux pas ajouter le fichier XML de 1,3Mo dans les sources, alors en attendant de l'élaguer je commence le test par un téléchargement du XML en question.

Je vais me creuser la tête pour le support des deux autres formats.

#7 - 17 octobre 2017 10:09 - Benjamin Dauvergne

- il faut un test de SSO dans le cadre du chargement d'une fédération, génère un fichier de métadonnées minimal (tu peux prendre celui de renater dans lequel tu ne garde que 3 IdPs), sans ça tu ne sais pas si ça marche de bout en bout, et ici clairement j'en doute, par exemple le code d'authent SAML dans authentic utilise get_idps() comme cela:

  ```
  src/authentic2_auth_saml/auth_frontends.py:        return app_settings.enable and list(get_idps())
  ```

  idem le code d'initiation d'un SSO ne va pas fonctionner et en fait tous les appels à self.get_idp(...) dans views.py
  Il va falloir que d'une manière ou d'une autre on puisse avoir la liste des IdPs;
- remplace le téléchargement du fichier Renater par ce fichier simplifié ;
- py.test fournit une fixture tmpdir qui est nettoyée automatiquement (c'est pratique ça évite de gérer ça) :
  https://docs.pytest.org/en/latest/tmpdir.html

Ce n'est pas un ticket évident en fait, sinon je l'aurai déjà fait :)

#8 - 17 octobre 2017 10:10 - Benjamin Dauvergne

Je pense qu'il va falloir nécessairement parcourir soit lasso.Server().providers soit directement le XML pour avoir dans un coin une liste des IdPs disponibles, ensuite MELLON_IDENTITY_PROVIDERS permet de définir pour les IdPs toute sorte de paramétrage ce serait bien de conserver cette possibilité (ERROR_URL, ERROR_REDIRECT_AFTER_TIMEOUT, etc..).

#9 - 19 octobre 2017 11:18 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

Merci pour les infos.
J'ai commencé par le plus facile, à savoir l'utilisation d'un fichier de fédération simplifié, contenant 4 IdPs.
Je pose le patch WIP ici, je m'attaque au reste.

#10 - 14 novembre 2017 10:42 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

Je suis en train de bosser sur la logique de prise en charge des trois formats possibles pour le chargement de fédération.

J'ai besoin de lire de la doc sur la mise en cache dans Django, je reviens ici dès que j'en sais un peu plus.

En attendant je pose le patch WIP avec un peu de progrès dans la fonction mellon.utils.create_server().

#11 - 14 novembre 2017 11:36 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

Commencé à jouer un peu avec le cache, patch WIP joint ici.
Je ne comprends toujours pas comment faire pour qu'une entrée du cache puisse se comporter en tant que fichier vis-à-vis de lasso.Server.loadMetadata() (argument const gchar *federation_file dans la doc pointée par Benj).
En l'occurrence l'entrée du cache est fedmd dans :

```
                        response = urllib2.urlopen(federation)
                        cache.get_or_set(fedmd, response.read(), 600)
```

lorsque les métadonnées de la fédération sont déclarées à l'aide d'une URL.

#12 - 14 novembre 2017 16:07 - Benjamin Dauvergne

Pour le cache je ne pensais pas à utiliser l'infrastructure de cache de Django, les fichiers de métadonnées font plusieurs méga-octets c'est un peu trop, mais plutôt utiliser django.core.storage (de plus get_or_set() n'est pas disponible en Django 1.8, on est pas encore passé en Django 1.11).

Le pseudo code ce serait ça (à valider, et faut gérer le cas où le fichier de fédération n'est pas encore chargé, avec un message):

```
import stat
import fnctl
import tempfile
import threading
from datetime import timedelta

from django.utils.timezone import now
from django.core.files import ContentFile

import requests
from pytz import utc, timezone
from datetime import datetime
from time import mktime
import os
import hashlib
import os.path

from django.core.files.storage import default_storage

def truncate_unique(s, length):
    if len(s) < length:
        return s
    md5 = hashlib.md5(s.encode('ascii')).hexdigest()
    # on tronque par le milieu parce que dans une URL le début et la fin sont intéressants
    l = (length - len(md5)) / 2 - 2
    assert l > 20 # en dessou c'est un peu court
    return s[:l] + '...' s[-l:] + '_' + md5

def dt_to_timestamp(dt):
    return mktime(utc.localize(input_date).utctimetuple())

def load_federation_cache(url):
    try:
        filename = truncate_unique(slugify(url), 250)
        path = os.path.join('metadata-cache', filename)

        unix_path = default_storage.path(path)
        f = open(unix_path, 'w')
        try:
            fcntl.lockf(f, fcntl.LOCK_EX | fnctl.LOCK_NB)
        except IOError:
            return
        else:
            try:
                # increase modified time by one hour to prevent too many updates
                st = stat.stat(unix_path).st_atime
                os.utime(unix_path, (atime, (st.st_atime, st.st_mtime + 3600))
                response = requests.get(url)
                response.raise_for_status()
```

```
                with tempfile.NamedTemporaryFile(dir=os.path.dirname(unix_path), delete=False) as temp:
                    temp.write(response.content)
                    temp.flush()
                    os.rename(temp.name, unix_path)
            except:
                os.unlink(temp.name)
            finally:
                fcntl.lockf(f, fcntl.LOCK_UN)
        finally:
            f.close()
    except:
        logging.exception(u'failed to load federation from %s', url)

def get_federation_from_url(url, update_cache=False):
    filename = truncate_unique(slugify(url), 250)
    path = os.path.join('metadata-cache', filename)
    if default_storage.exists(path):
        if default_storage.get_created_time(path) < now() - timedelta(days=1):
            threading.Thread(target=load_federation_cache, args=(url,)).start()
        return default_storage.open(path).read()
    logging.warning('federation %s has not been loaded', url)
    return None
```

#13 - 16 novembre 2017 16:28 - Paul Marillonnet

Ok je vais me baser là-dessus, merci.
Est-ce que j'inclus tout ça dans mellon.utils, ou bien je fais un module à part du genre mellon.federation_utils ?

#14 - 16 novembre 2017 16:39 - Paul Marillonnet

Je comprends pas très bien le comportement de get_federation_from_url.
Est-ce qu'on ne souhaite pas chercher des métadonnées si celles si ne sont pas déjà en cache ?

Je serais tenté de changer

```
    if default_storage.exists(path):
        if default_storage.get_created_time(path) < now() - timedelta(days=1):
```

en

```
    if not default_storage.exists(path) or \
            default_storage.get_created_time(path) < now() - timedelta(days=1):
```

Je me plante ?

#15 - 17 novembre 2017 16:36 - Paul Marillonnet

- Fichier 0001-WIP-add-federation_utils-module-19396.patch ajouté

Voilà j'ai inclus ton code à l'identique à peu de choses près, je vais bosser sur la fonctionnalité en elle-même maintenant.
Merci pour ton aide.

#16 - 17 novembre 2017 16:49 - Benjamin Dauvergne

Le code étant bien tordu faut vraiment des tests de tout ça en mockant le get() vers Renater.

#17 - 22 novembre 2017 12:19 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

Bon voilà ajouté le code et quelques tests, je viendrai compléter les tests au fur et à mesure.

#18 - 23 novembre 2017 15:23 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

Je suis en train d'implémenter le support des trois types de formats proposés ici.
J'ai encore des erreurs de chargement de données lors d'une config avec un fichier local, sans certificat X.509, je vais regarder d'où ça vient, je pose un patch WIP en attendant.

#19 - 24 novembre 2017 12:28 - Paul Marillonnet

Plus d'erreur de chargement détectée côté lasso, mais les IdPs ne sont pas enregistrés, je regarde d'où ça vient.

#20 - 24 novembre 2017 15:28 - Paul Marillonnet

Je suis en train de me creuser la tête pour tester correctement le format tuple ('fichier de metadonnées', 'certificat') mais je n'arrive pas à avoir sous la main un jeu de données de test valides.
En particulier, j'ai réduit le nombre d'IdP dans le fichier de md comme discuté plus haut, mais du coup le digest du fichier xml n'est plus correct (balise <ds:DigestValue>...</ds:DigestValue>).
J'ai une erreur lasso lors de la validation des métadonnées :

```
    def test_load_federation_tuple(mocker, rf, private_settings, caplog, tmpdir):
        private_settings.MELLON_FEDERATIONS = [
                ('tests/dummy_md.xml', 'tests/dummy_cert.pem')
        ]
        request = rf.get('/')
        assert 'failed with error' not in caplog.text
        with HTTMock(html_response):
            server = create_server(request)
>       assert len(server.providers) == 3
E       assert 0 == 3
E        +  where 0 = len(dict_proxy({}))
E        +    where dict_proxy({}) = <lasso.Server object at 0x7f620c8a9b10>.providers

tests/test_utils.py:79: AssertionError
---------------------------------------------------------------- Captured log call -----------------------------
-----------------------------------
utils.py                     95 INFO     Loading local cert-based federation ('tests/dummy_md.xml', 'tests/dumm
y_cert.pem')
lasso.py                   4346 DEBUG    func=xmlSecOpenSSLEvpDigestVerify:file=digests.c:line=305:obj=sha256:s
ubj=unknown:error=12:invalid data:data and digest do not match

utils.py                    118 ERROR    bad metadata for federation ('tests/dummy_md.xml', 'tests/dummy_cert.p
em')
utils.py                    119 DEBUG    lasso error: <lasso.DsSignatureVerificationFailedError(-111): Failed t
o verify signature.>
```

Je vais chercher à regénérer une empreinte sha256 valide pour le fichier de données de test.

#21 - 24 novembre 2017 15:46 - Paul Marillonnet

J'ai l'impression que l'utilitaire xmlsec1 ne permet pas de regénérer simplement le contenu de DigestValue, mais seulement l'intégralité de la signature d'un document XML.
Est-ce que ça signifie que, pour tester le format tuple, il faut que je crée une paire de clés de signature pour une fédération factice, et que je génère le certificat X.509 associé ?

#22 - 24 novembre 2017 15:58 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

Bon je lâche temporairement l'affaire sur cette histoire de tests du format tuple, je veux avancer sur le reste.
Je pose mon patch WIP des dernières modifs, avec les tests qui passent.

#23 - 24 novembre 2017 16:27 - Paul Marillonnet

Benjamin Dauvergne a écrit :

> Je pense qu'il va falloir nécessairement parcourir soit lasso.Server().providers soit directement le XML pour avoir dans un coin une liste des IdPs disponibles

Je n'arrive pas à retrouver parmi les attributs de lasso.Provider les champs correspondant au contenu des dicos renvoyés par le générateur adapters.DefaultAdapter().get_idps.
Donc je vais plutôt partir sur l'option 2, à savoir parcourir directement le XML pour reconstruire la liste la IdPs disponibles (même si en théorie cette option me paraît moins élégante que l'option 1).

#24 - 24 novembre 2017 16:39 - Benjamin Dauvergne

Ne teste pas la signature des métadonnées, tu pourras avancer comme cela. Comme pour PSL ça n'a pas grand intérêt.

#25 - 24 novembre 2017 16:40 - Benjamin Dauvergne

Paul Marillonnet a écrit :

> Benjamin Dauvergne a écrit :

> Je pense qu'il va falloir nécessairement parcourir soit lasso.Server().providers soit directement le XML pour avoir dans un coin une liste des IdPs disponibles

> Je n'arrive pas à retrouver parmi les attributs de lasso.Provider les champs correspondant au contenu des dicos renvoyés par le générateur adapters.DefaultAdapter().get_idps.
> Donc je vais plutôt partir sur l'option 2, à savoir parcourir directement le XML pour reconstruire la liste la IdPs disponibles (même si en théorie cette option me paraît moins élégante que l'option 1).

Lasso n'extrait que très peu de choses, il faut effectivement mieux parcourir le XML en python.

#26 - 24 novembre 2017 18:06 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

J'implémente parcours le XML dans mellon.adapters.DefaultAdapter().get_identity_providers_setting, comme dans le patch joint.
Ca m'a l'air d'être la solution la plus simple, puisque cette méthode est ensuite appelée par mellon.adapters.DefaultAdapter().get_idps et mellon.adapters.DefaultAdapter().get_idp, et donc aussi par mellon.utils.get_idps et mellon.utils.get_idp.

#27 - 24 novembre 2017 18:26 - Paul Marillonnet

Les tests mellon passent avec le patch. Je vais tester côté authentic.

#28 - 24 novembre 2017 18:37 - Paul Marillonnet

Côté A2 les tests aussi passent.
Hormis un jeu de tests un peu plus complet (que je viendrai ajouter plus tard, dans le prochain patch), est-ce que j'oublie quelque chose (si on part du principe qu'on ne vérifie pas la signature des métadonnées) ?

#29 - 24 novembre 2017 19:04 - Paul Marillonnet

En fait oui, j'oublie de vérifier que les IdPs renvoyés par get_idps à partir du XML ont effectivement été chargés dans lasso.Server().providers.
Il faut aussi que j'implémente les paramétrages similaires à ce qui est fait pour MELLON_IDENTITY_PROVIDERS.

#30 - 25 novembre 2017 09:58 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

Patch à jour, avec le retrait de l'appel inutile à mellon.utils.get_federations dans mellon.utils.create_server, et la prise en charge des paramétrages similaires à MELLON_IDENTITY_PROVIDERS.

#31 - 25 novembre 2017 11:43 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

Paul Marillonnet a écrit :

> En fait oui, j'oublie de vérifier que les IdPs renvoyés par get_idps à partir du XML ont effectivement été chargés dans lasso.Server().providers.

Je crois maintenant que je me plantais ici, ce n'est pas le comportement attendu.
get_idps est d'ailleurs utilisé en amont de la phase de chargement des IdPs, donc il ne faut pas inspecter la liste lasso.Server().providers, qui de toute façon sera vide à ce moment là.

Patch à jour, vérification de la signature supprimée comme conseillé par Benjamin. J'ai aussi mis à jour le fichier README.

Je voulais définir en extension l'ensemble des paramètres de fédération pour MELLON_FEDERATIONS, mais je pense que c'est mieux de laisser la possibilité de redéfinir localement, pour chaque fédération, tout paramètre global à mellon.
Pas certain que ce soit nécessaire, et je ne vois pas dans le code existant une telle redéfinition, qui serait faite pour n'importe quelle clé fournie dans un élément de MELLON_IDENTITY_PROVIDERS (alors que c'est ce que le README, à la section MELLON_IDENTITY_PROVIDERS laisse entendre).
Mais c'est plus simple que de lister explicitement les paramètres supportés.

Il faut encore que je teste des cas plus complexes, avec notamment des mélanges de config MELLON_IDENTITY_PROVIDERS et MELLON_FEDERATIONS.

#32 - 25 novembre 2017 11:48 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

Mal relu mon précédent patch, oublié de supprimer deux lignes de test du cache django dans mellon.utils (lequel n'est finalement pas utilisé), c'est corrigé ici.

#33 - 27 novembre 2017 00:26 - Benjamin Dauvergne

Le get_identity_providers_setting() m'a l'air un peu violent, dans un cas réel ça va charger un fichier de 20Mo et le parser à chaque appel.

Il faudra trouver un moyen:
1. de faire du cache,
2. de ne pas garder 20Mo de métadonnées en mémoire en permanence, et donc stocker ça en base ou sur disque (on parse le fichier de fédération, et on réécrit ensuite chaque fichier de métadonnée un par un) et ne charger les métadonnées qu'à la demande.

Dans la mesure où ce code n'est pas sensé allé en production mais est uniquement pour le POC Condorcet ça ira, mais faut garder en mémoire que c'est pas prod-proof si on a d'autres commandes dans l'univers des fédérations universitaires.

#34 - 27 novembre 2017 10:45 - Paul Marillonnet

Ok oui c'est vrai.
Je pense réutiliser en partie la procédure de mise en cache des fichiers de fédérations pour les métadonnées spécifiques aux IdPs qui y sont contenus.
Peut-être qqchose comme :

```
def idp_metadata_filepath(entity_id):
    filename = truncate_unique(slugify(entity_id), 250)
    return os.path.join('metadata-cache', filename)

def idp_metadata_needs_refresh(entity_id, update_cache=False):
    filepath = idp_metadata_filepath(entity_id)
    if not default_storage.exists(filepath) or update_cache or \
            default_storage.created_time(filepath) < datetime.now() - timedelta(days=1):
        return True
    return False

def idp_metadata_store(entity_id, metadata):
    logger = logging.getLogger(__name__)
    if not idp_metadata_needs_refresh(entity_id):
        return
    filepath = idp_metadata_filepath(entity_id)
    with open(filepath, 'w') as f:
        try:
            fcntl.lockf(f, fcntl.LOCK_EX | fcntl.LOCK_NB)
            f.write(metadata)
            f.close()
            fcntl.lockf(f, fcntl.LOCK_UN)
        except:
            logger.error('Couldn\'t store metadata for EntityID %r',
                    entity_id)

def idp_metadata_load(entity_id):
    logger = logging.getLogger(__name__)
    filepath = idp_metadata_filepath(entity_id)
    if default_storage.exists(filepath):
        logger.info('Loading metadata for EntityID %r', entity_id)
        with open(filepath, 'r') as f:
            return f.read()
```

Il me semble que MELLON_IDENTITY_PROVIDERS prend en charge des fichiers locaux, je pense que la mise en cache dont tu parles devrait réutiliser cette fonctionnalité déjà existante.

#35 - 27 novembre 2017 12:22 - Benjamin Dauvergne

Je ne sais pas trop, ça m'embête un peu de matérialiser toutes ces métadonnées sous forme de configuration de django-mellon, le moins on charge en mémoire le mieux c'est pour moi.

Il y a un souci là non:

```
    return path
    logger.warning('federation %s has not been loaded', url)
    return None
```

aussi l'objet LassoServer est mis en cache à sa première utilisation, on va forcément tomber sur le cas ou la fédération n'est pas en cache on crée l'objet lasso.Server() et puis ce n'est jamais plus chargé.

Idéalement on ne devrait charger que la métadonnée du fournisseur qui nous intéresse, et ne plus mettre en cache lasso.Server() (qui ne coûte pas si cher à créer en fait).
L'approche pour cela est celle faite dans authentic2, après un login.processAuthnResponseMsg(), login.processResponseMsg(), logout.processRequestMsg() ou un logout.processResponseMsg, on intercepte les exceptions lasso.ProfileUnknownProviderError ou lasso.ServerProviderNotFoundError, et on tente de charge le fournisseur dont l'entityID est dans login/logout.remoteProviderId, si au deuxième coup ça foire,

#36 - 27 novembre 2017 15:11 - Paul Marillonnet

Benjamin Dauvergne a écrit :

> Je ne sais pas trop, ça m'embête un peu de matérialiser toutes ces métadonnées sous forme de configuration de django-mellon, le moins on charge en mémoire le mieux c'est pour moi.

Je voulais que mellon ait connaissance, pour chaque IdP, d'un fichier de cache où aller chercher les métadonnées le concernant. Donc peut-être une approche doublement "lazy", car :
1) évitant de charger toutes les métadonnées en mémoire (ce qui en effet va dans le sens de l'utilisation get_idps() dans le frontend d'authentification SAML d'A2, servant simplement de variable booléenne).
2) et peut-être en réutilisant la fonctionnalité de mention d'un chemin local dans le champ 'METADATA' d'un dictionnaire d'IdP (au lieu de faire appaître dans ce champ 'METADATA' la totalité du contenu du descripteur d'entité SAML)

Paul Marillonnet a écrit :

> Il faut encore que je teste des cas plus complexes, avec notamment des mélanges de config MELLON_IDENTITY_PROVIDERS et MELLON_FEDERATIONS.

En fait l'idée de départ était qu'une fois tout le boulot de chargement/configuration des IdPs effectué il n'y ait plus de distinction, dans le code de mellon, entre IdPs issus d'un fichier de fédération et IdPs mentionnés par MELLON_IDENTITY_PROVIDERS. Mais peut-être que la façon dont va être utilisée cette nouvelle fonctionnalité ne requiert pas ça (i.e. pas de cas de mélanges des deux configs MELLON_IDENTITY_PROVIDERS et MELLON_FEDERATION) ?

> Il y a un souci là non:

Aïe. Merci, c'est corrigé.

> aussi l'objet LassoServer est mis en cache à sa première utilisation, on va forcément tomber sur le cas ou la fédération n'est pas en cache on crée l'objet lasso.Server() et puis ce n'est jamais plus chargé.
>
> Idéalement on ne devrait charger que la métadonnée du fournisseur qui nous intéresse, et ne plus mettre en cache lasso.Server() (qui ne coûte pas si cher à créer en fait).
> L'approche pour cela est celle faite dans authentic2, après un login.processAuthnResponseMsg(), login.processResponseMsg(), logout.processRequestMsg() ou un logout.processResponseMsg, on intercepte les exceptions lasso.ProfileUnknownProviderError ou lasso.ServerProviderNotFoundError, et on tente de charge le fournisseur dont l'entityID est dans login/logout.remoteProviderId, si au deuxième coup ça foire,

Ok je vais regarder ça. Merci.

#37 - 28 novembre 2017 08:50 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

#39 - 28 novembre 2017 11:42 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

Bon, je voulais aller au bout du patch précédent pour voir ce que ça donnerait "en vrai", avec les tests mellon qui passent.
(Je joins le patch ici, même si sans doute c'est un flagrant délit de "code churn" à outrance, avec sûrement des ajouts pas pertinents).

#40 - 28 novembre 2017 12:25 - Benjamin Dauvergne

Test tests sont trop unitaires pour tester quoi que ce soit, il faut un test de bout en bout sur un login et on verra que le cache ne marche pas sans ce que je décris plus haut.

#41 - 30 novembre 2017 15:20 - Paul Marillonnet

Ok d'ac je vais écrire ce test, en piochant dans les portions de procédures de login déjà testées dans mellon.
J'ai l'impression que je pourrais aussi m'inspirer de ce qui est fait dans authentic2_auth_saml.auth_frontend.SAMLFrontend.login()

#42 - 30 novembre 2017 16:37 - Paul Marillonnet

- Fichier auth_samlresponse.xml ajouté

J'imagine qu'il faut fournir à mellon les parties du dialogue SAML pour un SP et un IDP "mockés", en plus de mimer le comportement d'un client.
Peut-être par exemple, comme dans le fichier ci-joint, une réponse SAML de l'IdP avec les attributs renvoyés dans l'assertion, pour simuler un login effectué avec succès ?

#43 - 30 novembre 2017 17:59 - Benjamin Dauvergne

Dans test_sso_slo.py tu as une classe qui mocke un IdP le souci ici ça va être de mocker un IdP venant de ton fichier de fédération, le plus simple c'est d'ajouter les métadonnées qui sont déjà dans les tests dans ton fichier de fédération de test.

#44 - 30 novembre 2017 19:26 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

Ok, j'ai ajouté les métadonnées dans le fichier de fédération.
Je suis d'ajouter un test de login complet avec chargement du fichier de fédération (dans le patch WIP joint, fichier @test_sso_slo.py).

#45 - 01 décembre 2017 11:08 - Paul Marillonnet

J'ai corrigé le idp no défini en federated_idp dans le code de la fonction de test ajoutée dans test_sso_slo.py.
Et en effet ça s'avère subtil une fois qu'on a un test de login complet avec fichier de fédération chargé.

Le test renvoie une trace, je vais regarder ce qui ne va pas :

```
================================================ FAILURES ================================================
_______________________________________________ test_login _______________________________________________

db = None, app = <django_webtest.DjangoTestApp object at 0x7ff420477cd0>
federated_idp = <test_sso_slo.MockFederatedIdp object at 0x7ff42014ab10>
caplog = <_pytest.logging.LogCaptureFixture object at 0x7ff41fff2290>
federated_sp_settings = <django.conf.LazySettings object at 0x7ff421c04790>

    def test_login(db, app, federated_idp, caplog, federated_sp_settings):
>       response = app.get(reverse('mellon_login'))

tests/test_sso_slo.py:204: 
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
/tmp/tox-paul/django-mellon/coverage-dj18-sqlite/local/lib/python2.7/site-packages/django_webtest/__init__.py:
124: in get
    response = super(DjangoTestApp, self).get(url, **kwargs)
/tmp/tox-paul/django-mellon/coverage-dj18-sqlite/local/lib/python2.7/site-packages/webtest/app.py:331: in get
    expect_errors=expect_errors)
/tmp/tox-paul/django-mellon/coverage-dj18-sqlite/local/lib/python2.7/site-packages/django_webtest/__init__.py:
87: in do_request
    expect_errors)
/tmp/tox-paul/django-mellon/coverage-dj18-sqlite/local/lib/python2.7/site-packages/webtest/app.py:651: in do_r
equest
    self._check_status(status, res)
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

self = <django_webtest.DjangoTestApp object at 0x7ff420477cd0>, status = None
res = <400 BAD REQUEST text/html body='no idp found'>

    def _check_status(self, status, res):
        if status == '*':
            return
        res_status = res.status
        if (isinstance(status, string_types) and '*' in status):
            if re.match(fnmatch.translate(status), res_status, re.I):
                return
        if isinstance(status, string_types):
            if status == res_status:
                return
        if isinstance(status, (list, tuple)):
            if res.status_int not in status:
                raise AppError(
                    "Bad response: %s (not one of %s for %s)\n%s",
                    res_status, ', '.join(map(str, status)),
                    res.request.url, res)
            return
        if status is None:
            if res.status_int >= 200 and res.status_int < 400:
                return
            raise AppError(
                "Bad response: %s (not 200 OK or 3xx redirect for %s)\n%s",
                res_status, res.request.url,
>               res)
E           AppError: Bad response: 400 BAD REQUEST (not 200 OK or 3xx redirect for http://testserver/login/)
E           no idp found

/tmp/tox-paul/django-mellon/coverage-dj18-sqlite/local/lib/python2.7/site-packages/webtest/app.py:683: AppErro
```

```
r
================================== 1 failed, 4 passed in 2.41 seconds ==================================
ERROR: InvocationError: '/tmp/tox-paul/django-mellon/coverage-dj18-sqlite/bin/py.test --random --junit-xml=jun
it-coverage-dj18-sqlite.xml --cov=mellon --cov-report xml tests/test_sso_slo.py'
```

#46 - 01 décembre 2017 11:48 - Paul Marillonnet

C'était une erreur de config de ma part.
Maintenant c'est une ServerProviderNotFoundError, je vais déboguer pour voir d'où ça vient, mais clairement le code tel que je l'ai écrit pour l'instant ne fonctionne pas.
Je vais regarder comment je pourrais implémenter dans mellon ce que tu décris plus haut (la double capture d'exception avec un rejeu du chargement de l'IdP).

#47 - 04 décembre 2017 15:41 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

Comportement étrange. J'ai besoin d'aller voir ce qui se passe dans le code de lasso pour comprendre ce qui ne va pas dans mon code.
Les métadonnées de l'IdP sont bien présentes, comme pour les trois autres du fichier de fédération, dans le frozendict providers du serveur lasso chargé.
Et pourtant cette erreur qui laisse croire le contraire :

```
_____________________________________________________________________________ test_login_federatio
n_chosen_idp _____________________________________________________________________________

db = None, app = <django_webtest.DjangoTestApp object at 0x7fc11d2900d0>, federated_idp = <test_sso_slo.MockFe
deratedIdp object at 0x7fc11cf63e10>
caplog = <_pytest.logging.LogCaptureFixture object at 0x7fc11ceb2050>, federated_sp_settings = <django.conf.La
zySettings object at 0x7fc11ea1b7d0>

    def test_login_federation_chosen_idp(db, app, federated_idp, caplog, federated_sp_settings):
        qs = urlencode({
            'entityID': 'http://idp5/metadata',
            'next': '/whatever',
        })
        response = app.get('/login/?' + qs)
>       url, body = federated_idp.process_authn_request_redirect(response['Location'])

tests/test_sso_slo.py:217:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
tests/test_sso_slo.py:84: in process_authn_request_redirect
    login.processAuthnRequestMsg(url.split('?', 1)[1])
/tmp/tox-paul/django-mellon/coverage-dj18-sqlite/local/lib/python2.7/site-packages/lasso.py:3516: in processAu
thnRequestMsg
    Error.raise_on_rc(rc)
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

rc = -201

    @staticmethod
    def raise_on_rc(rc):
        global exceptions_dict
        if rc != 0:
            exception = exceptions_dict.get(rc, Error())
            exception.code = rc
>           raise exception
E           ServerProviderNotFoundError: <lasso.ServerProviderNotFoundError(-201): The identifier of a provide
r is unknown to #LassoServer. To register a provider in a #LassoServer object, you must use the methods lasso_
server_add_provider() or lasso_server_add_provider_from_buffer().>

/tmp/tox-paul/django-mellon/coverage-dj18-sqlite/local/lib/python2.7/site-packages/lasso.py:62: ServerProvider
NotFoundError
```

Je pose ici le patch WIP, avec quelques corrections dans la procédure de test de login complet.

#48 - 04 décembre 2017 15:57 - Benjamin Dauvergne

Tu peux me sortir un print login.server.providers au point d'exception dans login.processAuthnRequestMsg (lancer py.test avec l'option --pdb puis remonter 2 fois dans la trace avec la commande en ligne up de pdb).

#49 - 04 décembre 2017 16:32 - Paul Marillonnet

Voilà, l'IdP mocké y est aussi :

```
tests/test_sso_slo.py F
                                                                                                   [100%]
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> tracebac
k >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

db = None, app = <django_webtest.DjangoTestApp object at 0x7f5b079c6450>, federated_idp = <test_sso_slo.MockFe
deratedIdp object at 0x7f5b075d4790>
caplog = <_pytest.logging.LogCaptureFixture object at 0x7f5b075223d0>, federated_sp_settings = <django.conf.La
zySettings object at 0x7f5b0908c7d0>

    def test_login_federation_chosen_idp(db, app, federated_idp, caplog, federated_sp_settings):
        qs = urlencode({
            'entityID': 'http://idp5/metadata',
            'next': '/whatever',
        })
        response = app.get('/login/?' + qs)
>       url, body = federated_idp.process_authn_request_redirect(response['Location'])

tests/test_sso_slo.py:208: 
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
tests/test_sso_slo.py:75: in process_authn_request_redirect
    login.processAuthnRequestMsg(url.split('?', 1)[1])
/tmp/tox-paul/django-mellon/coverage-dj18-sqlite/local/lib/python2.7/site-packages/lasso.py:3516: in processAu
thnRequestMsg
    Error.raise_on_rc(rc)
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

rc = -201

    @staticmethod
    def raise_on_rc(rc):
        global exceptions_dict
        if rc != 0:
            exception = exceptions_dict.get(rc, Error())
            exception.code = rc
>           raise exception
E           ServerProviderNotFoundError: <lasso.ServerProviderNotFoundError(-201): The identifier of a provide
r is unknown to #LassoServer. To register a provider in a #LassoServer object, you must use the methods lasso_
server_add_provider() or lasso_server_add_provider_from_buffer().>

/tmp/tox-paul/django-mellon/coverage-dj18-sqlite/local/lib/python2.7/site-packages/lasso.py:62: ServerProvider
NotFoundError
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> entering P
DB >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
> /tmp/tox-paul/django-mellon/coverage-dj18-sqlite/local/lib/python2.7/site-packages/lasso.py(62)raise_on_rc()
-> raise exception
(Pdb) up
> /tmp/tox-paul/django-mellon/coverage-dj18-sqlite/local/lib/python2.7/site-packages/lasso.py(3516)processAuth
nRequestMsg()
-> Error.raise_on_rc(rc)
(Pdb) up
> /home/paul/eodevel/django-mellon/tests/test_sso_slo.py(75)process_authn_request_redirect()
-> login.processAuthnRequestMsg(url.split('?', 1)[1])
(Pdb) print login.server.providers
frozendict({'https://ambre.vetagro-sup.fr/idp/shibboleth': <lasso.Provider object at 0x7f5b074909d0>, 'https:/
/antimoine.insa-strasbourg.fr/idp/shibboleth': <lasso.Provider object at 0x7f5b07490190>, 'https://aishib.agro
polis.fr/idp/shibboleth': <lasso.Provider object at 0x7f5b074904d0>, 'https://access-check.edugain.org/simples
aml/saml2/idp/metadata.php': <lasso.Provider object at 0x7f5b07490110>, 'http://idp5/metadata': <lasso.Provide
r object at 0x7f5b07490210>})
```

#50 - 04 décembre 2017 21:29 - Benjamin Dauvergne

Tu as chargé les métadonnées de l'IdP dans l'IdP, c'est les métadonnées du SP qu'il faut charger dans l'IdP.

#51 - 05 décembre 2017 10:00 - Paul Marillonnet

Ok, je pensais que le code de tests imitait le comportement d'un login initié par le SP, et donc qu'on se situait ici côté SP dans le test -- et en tout cas qu'il ne s'agissait pas ici de "charger les métadonnées de l'IdP dans l'IdP".
Je trouvais ça légitime que mellon manipule côté SP un objet serveur lasso qui contiennent tous les IdPs enregistrés, càd les IdPs auprès desquels le SP peut réaliser le SSO.

Où est-ce que je me plante dans mon raisonnement ?

Enfin bref, le problème est entre ma chaise de bureau et mon clavier d'ordinateur. Je pense qu'en ayant un jeu de fixtures un peu plus clair, je serai en mesure de comprendre ce qui cloche dans mon code.

#52 - 05 décembre 2017 12:19 - Benjamin Dauvergne

Paul Marillonnet a écrit :

> Ok, je pensais que le code de tests imitait le comportement d'un login initié par le SP, et donc qu'on se situait ici côté SP dans le test -- et en tout cas qu'il ne s'agissait pas ici de "charger les métadonnées de l'IdP dans l'IdP".

La classe MockIdP, comme son nom l'indique, simule un IdP, la méthode processAuthnRequest est la méthode qu'un IdP appelle pour parser un requête d'authentification SAML en provenance d'un SP, quand tu m'afficher le login.server.providers à cet endroit je n'y vois que des IdPs alors qu'on ne devrait y voir que les métadonnés du fournisseur de service de test, celles fournies par la fixture "sp_metadata".

> Je trouvais ça légitime que mellon manipule côté SP un objet serveur lasso qui contiennent tous les IdPs enregistrés, càd les IdPs auprès desquels le SP peut réaliser le SSO.
> Où est-ce que je me plante dans mon raisonnement ?

Tu n'est pas dans django-mellon tu es dans la classe MockIdp des tests.

> Enfin bref, le problème est entre ma chaise de bureau et mon clavier d'ordinateur. Je pense qu'en ayant un jeu de fixtures un peu plus clair, je serai en mesure de comprendre ce qui cloche dans mon code.

En fait ce code est inutile, tu peux utiliser la fixture idp existante:

```
class MockFederatedIdp(MockIdp):
    def __init__(self, request):
        self.server = create_server(request)
```

La configuration de l'IdP ne change rien en fédération ou pas en fédération, c'est le SP qui change sa vision des choses (il voit pleins d'IdP chargés via un fichier unique au lieu de pleins d'IdP chargés via une configuration individuelle pour chacun d'entre eux).

#53 - 05 décembre 2017 17:03 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

Merci, c'est bien plus clair comme ça.

J'ai donc ce test de login :

```
def test_login_federation(db, app, federated_idp, caplog, federated_sp_settings):
    qs = urlencode({
        'entityID': 'http://idp5/metadata',
        'next': '/whatever',
    })
    response = app.get('/login/?' + qs)
    url, body = federated_idp.process_authn_request_redirect(response['Location'])
    assert url.endswith(reverse('mellon_login'))
    response = app.post(reverse('mellon_login'), params={'SAMLResponse': body})
    assert 'created new user' in caplog.text
    assert 'logged in using SAML' in caplog.text
    assert response['Location'].endswith(federated_sp_settings.LOGIN_REDIRECT_URL)
```

(Je joins le patch WIP actuel, avec les tests qui passent, et je vais me creuser la tête pour ajouter d'autres scénarios de test).

#54 - 07 décembre 2017 12:52 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

Commencé à ajouter des tests de login avec fédération chargée, en adaptant ceux existants.

#55 - 07 décembre 2017 14:30 - Paul Marillonnet

Je vais aussi ajouter des tests pour le support des paramètres de configuration par élément de MELLON_FEDERATIONS, redéfinissant les paramètres mellon globaux.

(Je suis pas satisfait du code pour cette fonctionnalité dans le patch WIP actuel, c'est un peu "au petit bonheur la chance", et il n'y a pas de contrôle

sur les éléments paramétrés à l'intérieur ce chaque dictionnaire de fédération. Je vais chercher à implémenter ça d'une façon plus réglo).

#56 - 14 décembre 2017 17:55 - Paul Marillonnet

J'aurais bien aimé avoir un test unique de la forme (ici dans tests/test_utils.py) :

```
def test_federation_parameters(mocker, rf, private_settings, caplog, tmpdir):
    private_settings.MELLON_FEDERATIONS = [{
            'FEDERATION': 'tests/federation-sample.xml',
            'VERIFY_SSL_CERTIFICATE': False,
            'ERROR_REDIRECT_AFTER_TIMEOUT': 150,
            'PROVISION': True
            # ...
    }]
    request = rf.get('/')
    assert 'failed with error' not in caplog.text
    with HTTMock(html_response):
        server = create_server(request)
    assert len(server.providers) == 5
    # assert on teste ici les settings
```

Mais j'ai l'impression qu'à ce moment on ne peut plus que tester sur les conséquences du paramétrage, et pas le paramétrage en lui-même.
Je regarde plus en détail côté interface lasso.

#57 - 21 décembre 2017 14:49 - Paul Marillonnet

Je ne sais pas si le cache devrait ou non prendre en compte ces paramètres de configuration par fédération.
Dans l'idéal j'aimerais bien avoir un code uniforme pour les providers chargés individuellement et ceux issus d'un fichier de fédération, mais je crois que ça implique de modifier le système de cache dans mellon.federation_utils dans l'état actuel du code dans le patch.

#58 - 21 décembre 2017 15:36 - Benjamin Dauvergne

Lapin compris, décris ce qui te reste à faire d'après toi, moi je vois la chose suivante:
à partir d'un entity_id (obtenu depuis /login?entityID= ou bien lors du traitement d'une requête ou réponse reçu, d'obtenir la configuration et les métadonnées que ça vienne d'une fédération ou de la configuration de base, i.e. il faut que get_idp(entity_id) fonctionne comme pout settings.IDENTITY_PROVIDER, dans en plus de retrouver les métadonnées depuis un entity_id il faut retrouver la définition de la fédération qui y correspond et pour cela je mettrai la configuration extrait de settings dans un fichier json à coté du fichier de métadonnées (même nom de fichier et on rajouter _settings.json).

#59 - 21 décembre 2017 17:34 - Paul Marillonnet

Je me pose peut-être des problèmes qui n'en sont pas vraiment, mais avec ce que tu proposes j'ai l'impression qu'il faut encore préciser le comportement pour la configuration d'un IdP déclaré dans deux fichiers de fédérations chargés indépendamment (et donc avec possiblement des configurations différentes).

Est-ce que c'est un cas anecdotique qui théoriquement n'est pas pris en charge mais qui ne risque pas de se produire ?
Ou bien est-ce qu'il faut ajouter dans le getter la possibilité de choisir la fédération de provenance (pour que l'IdP soit retrouvé avec la configuration voulue, i.e. celle de la fédération choisie parmi toutes celles où cet IdP est déclaré) ?

#60 - 22 décembre 2017 00:58 - Benjamin Dauvergne

Anecdotique, si un IdP se trouve dans deux fédérations différentes et qu'on charge les deux, je dirais que pour l'instant on s'en fout un peu.

#61 - 22 décembre 2017 09:11 - Paul Marillonnet

Ok (dans ce cas, si on part du principe qu'on crée un fichier par fournisseur, avec suffixe _settings.json sur le truncate_unique de l'EntityID du fournisseur, c'est la dernière fédération chargée qui imposera sa configuration).

Je vais aussi réduire la taille maximale de retour de truncate_unique, parce que si on rajoute encore les 14 caractères du suffixe on dépasse possiblement la taille maximale autorisée de fichier supportée par le système de fichiers (comme l'indique ton lien posté plus tôt).

#62 - 22 décembre 2017 12:00 - Benjamin Dauvergne

Moi tout me va, le plus important c'est que l'API suive, i.e. que get_idp() cache l'implémentation, si plus tard on veut tout refaire avec des modèles, le code ne doit pas bouger et get_idp() pourrait renvoyer une classe MellonIdP d'ailleurs, plutôt qu'un dico, qui aurait une méthode MellonIdP.create_server() etc.. mais bon ça dépendra de ton enthousiasme.

#63 - 05 janvier 2018 10:59 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

Déjà une première version avec ajout de la propagation des paramètres de fédération vers les fournisseurs (cf les fonctions idp_settings_* dans mellon.federation_utils, et l'usage de idp_settings_{store,load} dans mellon.adapters.DefaultAdapter).

Je pose ici un premier patch sans test pour donner une idée de ma façon de procéder, et je commence à écrire des tests que j'inclurai dans le patch

(avec les possibles corrections induites).

#64 - 05 janvier 2018 16:18 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

Voilà, un patch qui cette fois-ci fonctionne, avec un test de propagation des paramètres de fédération dans chaque IdP issu de cette fédération.

#65 - 08 janvier 2018 14:27 - Paul Marillonnet

Benjamin Dauvergne a écrit :

> Moi tout me va, le plus important c'est que l'API suive, i.e. que get_idp() cache l'implémentation, si plus tard on veut tout refaire avec des modèles, le code ne doit pas bouger et get_idp() pourrait renvoyer une classe MellonIdP d'ailleurs, plutôt qu'un dico, qui aurait une méthode MellonIdP.create_server() etc.. mais bon ça dépendra de ton enthousiasme.

Comment procéder pour que ça ne fasse pas doublon avec le mécanisme d'adapteurs déjà en place ?
Est-ce qu'il faudrait, par exemple et en plus de ce que tu proposes, déplacer dans les modèles les fonctionnalités offertes par les adapteurs, pour complètement vider le fichier adapters.py ?

#66 - 08 janvier 2018 15:25 - Benjamin Dauvergne

Ça me semble indépendant, get_idp() ça fait partie de l'interface entre le code Django (vues/backends) et les adapteurs, les adapteurs fournissent les implémentations.

#67 - 08 janvier 2018 16:41 - Paul Marillonnet

Un des avantages que je vois à cela c'est de pouvoir à tout moment accéder à la config d'un IdPs.
Donc peut-être une classe du genre :

```
class MellonIdP(models.Model):
    entity_id = models.TextField(
            verbose_name=_('Entity ID'),
            unique=True)
    metadata_url = models.URLField(
            verbose_name=_('Metadata URL'))
    metadata = models.FileField(
            verbose_name=_('Metadata file location'))
    verify_ssl_cert = models.BooleanField(
            verbose_name=_('Verify SSL certificate'))

    def __init__(self, *args, **kwargs):
        super(self, MellonIdP).__init__(*args, **kwargs)
        idp = kwargs.get('idp')
        if not isinstance(idp, dict):
            return

        for field, setting in self.Meta._fields_from_django_settings.items():
            if setting in idp:
                setattr(self, field, idp.get(setting))

    def create_server(self, *args, **kwargs):
        try:
            self.server = lasso.Server.addProviderFromBuffer(
                    lasso.PROVIDER_ROLE_IDP,
                    self.metadata)
        except lasso.Error as e:
            logger.error(u'bad metadata in idp %r', self.entity_id)
            logger.debug(u'lasso error: %s', e)
            self.server = None
        except IOError as e:
            logger.warning('No such metadata file: %r', self.metadata)
            self.server = None

    class Meta:
        verbose_name = _('Mellon Identity Provider')
        verbose_name_plural = _('Mellon Identity Providers')

        _fields_from_django_settings = {
                'entity_id': 'ENTITY_ID',
                'metadata': 'METADATA',
                'metadata_url': 'METADATA_URL',
                'verify_ssl_cert': 'VERIFY_SSL_CERTIFICATE',
```

```
                # ... TODO Handle full mellon settings parameters
        }
```

#69 - 09 janvier 2018 10:12 - Benjamin Dauvergne

Paul Marillonnet a écrit :

> Un des avantages que je vois à cela c'est de pouvoir à tout moment accéder à la config d'un IdPs.
> Donc peut-être une classe du genre :
> [...]

Ouaip mais non, faut différencier le POPO (Plain Old Python Object, équivalent du POJO en Java) et le modèle, mélanger les deux c'est se compliquer la vie.

#71 - 09 janvier 2018 15:22 - Benjamin Dauvergne

Arrêtez d'utiliser les notes privées.

J'ai appliqué le patch de mon coté tout me semble ok sauf que sur stretch j'ai un souci avec le test test_invalid_msg_on_artifact_resolve mais ça ne semble pas venir de tes modifications, plutôt de la version de Lasso dans stretch, je suis en train de refaire un paquet pour stretch quand ce sera bon je te dis quoi.

#72 - 09 janvier 2018 16:07 - Benjamin Dauvergne

Ack.

#73 - 09 janvier 2018 17:53 - Paul Marillonnet

- Statut changé de Nouveau à Résolu (à déployer)

```
commit 63993e360c89b22ea8e70fabf12d4f7e223cfb90
Author: Paul Marillonnet <pmarillonnet@entrouvert.com>
Date:   Tue Nov 14 10:36:44 2017 +0100

    support federation file loading (#19396)
```

#75 - 09 janvier 2018 21:19 - Frédéric Péters

- Lié à Bug #21067: Le commit de chargement d'un fichier de fédération casse le SSO Publik ajouté

#76 - 09 janvier 2018 21:45 - Benjamin Dauvergne

- Statut changé de Résolu (à déployer) à Nouveau

Arrêtez avec les notes privées.

J'ai reverté le commit.

#77 - 09 janvier 2018 22:02 - Frédéric Péters

En gros, de ce que j'ai commencé à rergarder pour débugger :

- get_idps acceptait des dictionnaires de la forme {'METADATA': '/chemin/vers/les/metadata'}, dans cette situation prenait lisait le fichier et remplaçait la clé METADATA par ce contenu,
  - ça pourrait peut-être être repris ainsi
- mais get_idp() avant itérait sur get_idps(), et le patch change ça pour plutôt itérer sur get_identity_providers_setting()
  - mais ça fait qu'on peut donc se trouver là avec {'METADATA': '/chemin/...'}, qu'on passe à côté du taf qui serait fait dans get_idps()
- toute l'histoire du cache de fédération, qui en gros va poser sur le disque les métadonnées tierces, n'a pas de sens quand on a déjà les métadonnées en local comme c'est le cas avec Publik.

Plus généralement il y a des passages qui laissent songeur, pourquoi donc charger le fichier pour en extraire un entity id qu'on a déjà :

```
+            metadata_content = idp_metadata_load(entity_id)
+            entity_id = idp_metadata_extract_entity_id(metadata_content)
```

#78 - 10 janvier 2018 04:08 - Paul Marillonnet

Frédéric Péters a écrit :

> - toute l'histoire du cache de fédération, qui en gros va poser sur le disque les métadonnées tierces, n'a pas de sens quand on a déjà les métadonnées en local comme c'est le cas avec Publik.

Ok là c'est moi qui ai mal compris le besoin, je pensais que regrouper toutes les métadonnées à un seul endroit était plus important que ne pas dupliquer celles-ci dans le cas d'un fichier local de md fournies à mellon.

> Plus généralement il y a des passages qui laissent songeur, pourquoi donc charger le fichier pour en extraire un entity id qu'on a déjà :

Oui en effet c'est maladroit de ma part. Je voulais distinguer un entity_id théorique fourni par l'application utilisatrice de mellon et un entity_id effectif extrait des métadonnées.

#79 - 10 janvier 2018 04:11 - Paul Marillonnet

Frédéric Péters a écrit :

> Plus généralement il y a des passages qui laissent songeur [...]

Je ne sais pas comment procéder pour proposer des correctifs du bogue et desdits passages.
Benj, puisque tu as annulé le commit, est-ce que je peux continuer à bosser dans le même patch ?
Ou bien je fais un ticket ?

#80 - 10 janvier 2018 08:07 - Frédéric Péters

> Ok là c'est moi qui ai mal compris le besoin, je pensais que regrouper toutes les métadonnées à un seul endroit était plus important que ne pas dupliquer celles-ci dans le cas d'un fichier local de md fournies à mellon.

Sur le besoin je ne peux pas vraiment me prononcer là-dessus, l'origine n'est pas Publik; et il peut y avoir un sens à dupliquer (genre assurer que toutes les métadonnées, qu'elles soient déclarées via Hobo/Publik ou un fichier de fédérations) se trouvent accessibles selon la même nomenclature).

> Je ne sais pas comment procéder pour proposer des correctifs du bogue et desdits passages.

Je suggérerais d'attendre une nouvelle relecture de Benj, qui donnera les bonnes directions.

#81 - 10 janvier 2018 08:40 - Frédéric Péters

> Je suggérerais d'attendre une nouvelle relecture de Benj, qui donnera les bonnes directions.

Aussi, en quittant un peu mon prisme "environnement de dev Publik", je suggérerais de commencer par l'ajout de tests (sso complet ?, juste get_idp() ?) couvrant la situation où un adapter fournit un get_identity_providers_setting retournant {"METADATA": "chemin"}.

#82 - 10 janvier 2018 09:51 - Paul Marillonnet

Frédéric Péters a écrit :

> > Je suggérerais d'attendre une nouvelle relecture de Benj, qui donnera les bonnes directions.
>
> Aussi, en quittant un peu mon prisme "environnement de dev Publik", je suggérerais de commencer par l'ajout de tests (sso complet ?, juste get_idp() ?) couvrant la situation où un adapter fournit un get_identity_providers_setting retournant {"METADATA": "chemin"}.

Oui OK.

En ayant aussi constaté le bug de mon côté cette nuit, je persiste à croire que l'absence de création du répertoire de métadonnées dans l'emplacement de l'appli lorsqu'on n'a pas recours à une fédération est en grosse partie en faute.

Est-ce que le seul moyen pour moi de tester mon code dans Publik est de disposer d'une installation complète en local ? (A ma connaissance on ne dispose pas de VMs Publik type "bac à sable" pour tester les patches, non ?)

#83 - 10 janvier 2018 10:25 - Thomas Noël

Paul Marillonnet a écrit :

> Est-ce que le seul moyen pour moi de tester mon code dans Publik est de disposer d'une installation complète en local ? (A ma connaissance on ne dispose pas de VMs Publik type "bac à sable" pour tester les patches, non ?)

En fait le bac à sable, c'est justement l'installation locale de Publik qu'il faut avoir en local sur sa machine.

#84 - 10 janvier 2018 10:53 - Paul Marillonnet

Thomas Noël a écrit :

> En fait le bac à sable, c'est justement l'installation locale de Publik qu'il faut avoir en local sur sa machine.

Je vais faire cette installation. Est-ce qu'on a une page wiki ou un début de doc là-dessus, que je pourrais consulter et compléter au fur et à mesure de mon installation ?

#85 - 10 janvier 2018 11:43 - Benjamin Dauvergne

Avant tout j'aimerai comprendre pourquoi les tests artifact / métadonnée classiques passent dans les tests avec ton patch mais ne marche pas dans l'environnement publik, c'est ça qu'il faut corriger.

Et donc ta remarque sur la réutilisation du répertoire MEDIA est la bonne, j'ai fait ce changement qui fait planter tous les tests, je te propose de l'intégrer et de corriger ce qui ne va pas en plus de simplifier le code:

```
commit bb51883515c4ba8b8799dfd3b4474b9363703140
Author: Benjamin Dauvergne <bdauvergne@entrouvert.com>
Date:   Wed Jan 10 11:40:34 2018 +0100

    wip

diff --git a/tests/test_sso_slo.py b/tests/test_sso_slo.py
index a5438d5..e40dfd8 100644
--- a/tests/test_sso_slo.py
+++ b/tests/test_sso_slo.py
@@ -38,7 +38,8 @@ def public_key():

 @fixture
-def sp_settings(private_settings, idp_metadata, sp_private_key, public_key):
+def sp_settings(request, private_settings, idp_metadata, sp_private_key, public_key, tmpdir):
+    private_settings.MEDIA_ROOT = str(tmpdir)
     private_settings.MELLON_IDENTITY_PROVIDERS = [{
         'METADATA': idp_metadata,
     }]
@@ -50,7 +51,8 @@ def sp_settings(private_settings, idp_metadata, sp_private_key, public_key):

 @fixture
-def federated_sp_settings(private_settings, federation_metadata, sp_private_key, public_key):
+def federated_sp_settings(request, private_settings, federation_metadata, sp_private_key, public_key, tmpdir)
:
+    private_settings.MEDIA_ROOT = str(tmpdir)
     private_settings.MELLON_FEDERATIONS = [{
         'FEDERATION': federation_metadata,
     }]
```

À reprendre mes remarques du début tu n'a pas implémenté le fonctionnement où on détecte d'abord l'entityID de la requête SAML reçue puis on charge les métadonnées comme dans authentic (avec la boucle autour de process*Request/ResponseMsg).

#86 - 10 janvier 2018 11:55 - Benjamin Dauvergne

Je reprendrai ton patch de manière à bien séparer le cas classique avec MELLON_IDENTITY_PROVIDERS qui ne doit pas changer de fonctionnement, et le nouveau avec fédération, la distinction entre les deux devant se faire uniquement dans get_idp() / get_idps() si on est dans l'ancien mode, aucune ligne de ton nouveau code ne devrait s'exécuter (pas de cache, rien).

#87 - 10 janvier 2018 18:53 - Paul Marillonnet

Benjamin Dauvergne a écrit :

> Et donc ta remarque sur la réutilisation du répertoire MEDIA est la bonne, j'ai fait ce changement qui fait planter tous les tests, je te propose de l'intégrer et de corriger ce qui ne va pas en plus de simplifier le code:

Oui OK je vais commencer par là.

> À reprendre mes remarques du début tu n'a pas implémenté le fonctionnement où on détecte d'abord l'entityID de la requête SAML reçue puis on charge les métadonnées comme dans authentic (avec la boucle autour de process*Request/ResponseMsg).

Il y a quelque chose qui m'échappe, je ne vois pas comment on pourrait se retrouver dans le cas que tu décris, où, si j'ai bien compris, mellon traite une requête issue d'un (ou à destination) d'un fournisseur qu'il n'a pas encore chargé en mémoire.
En fait, quand bien même lasso.Server() ne serait plus en cache, je ne comprends pas où dans le code il devient nécessaire de le créer à nouveau.

> Je reprendrai ton patch de manière à bien séparer le cas classique avec MELLON_IDENTITY_PROVIDERS qui ne doit pas changer de fonctionnement, et le nouveau avec fédération, la distinction entre les deux devant se faire uniquement dans get_idp() / get_idps() si on est dans l'ancien mode, aucune ligne de ton nouveau code ne devrait s'exécuter (pas de cache, rien).

Ok je n'avais pas compris, ça me paraissaît plus simple et plus pertinent d'implémenter un cache pour toutes les métadonnées quelles que soient leurs provenances.

#88 - 10 janvier 2018 19:48 - Paul Marillonnet

Paul Marillonnet a écrit :

> Il y a quelque chose qui m'échappe, je ne vois pas comment on pourrait se retrouver dans le cas que tu décris, où, si j'ai bien compris, mellon traite une requête issue d'un (ou à destination) d'un fournisseur qu'il n'a pas encore chargé en mémoire.
> En fait, quand bien même lasso.Server() ne serait plus en cache, je ne comprends pas où dans le code il devient nécessaire de le créer à nouveau.

Réponse de Mik en privé, je comprends maintenant ce qui ne va pas dans mon patch, dont l'implémentation de create_server ne tient pas compte de son usage dans l'appli :

```
  def create_login(request):
      server = create_server(request)
      [...]
```

Et

```
  def create_logout(request):

      server = create_server(request)
      [...]
```

Ce qui charge en mémoire à chaque fois l'intégralité des MDs alors qu'on traite possiblement une requête depuis/vers un seul IdP.

#89 - 11 janvier 2018 11:17 - Paul Marillonnet

Benjamin Dauvergne a écrit :

> Et donc ta remarque sur la réutilisation du répertoire MEDIA est la bonne, j'ai fait ce changement qui fait planter tous les tests, je te propose de l'intégrer et de corriger ce qui ne va pas en plus de simplifier le code:

Ce changement fait planter tous les tests à leur première exécution. Après, une fois que le dossier metadata-cache a été créé par les fonctions utilitaires de gestions des MDs de fédération, les tests réussissent. Il faudrait que mes tests ne placent pas les MDs dans le vrai dossier metadata-cache, mais dans un dossier factice écrasé à l'issue des tests (ce qui apparemment n'est pas encore le cas avec la modif que tu proposes, ayant recours à la fixture tmpdir).

Je me demande si d'ailleurs ça n'expliquerait pas l'échec du premier build Jenkins (1065) et le succès du second (1066) mardi soir. (Je ne sais pas où sont les scripts Jenkins pour le build de nos briques, mais peut-être que le serveur Jenkins n'a pas refait un git-clone lors du relancement manuel du build par Frédéric mardi soir, d'où le succès des tests au deuxième essai -- le répertoire metadata-cache étant cette fois-ci présent).
Ça expliquerait aussi pourquoi les tests unitaires pour le build en question (1065) plantait pour le premier des six virtualenv seulement (dans l'ordre chronologique d'exécution).

Et ayant pu appliquer le patch sur un VM sur laquelle tourne A2, j'ai pu effectuer un SSO complet après avoir chargé un fichier de fédération dans un premier temps, pour ensuite revenir à une config MELLON_IDENTITY_PROVIDERS de base.

Est-ce que, pour remédier à ça, on ne pourrait pas imaginer créer ce dossier de métadonnées à l'initialisation de l'appli, par exemple en plaçant une instruction os.makesdir(...) à la fin de mellon/__init__.py ?

#90 - 11 janvier 2018 11:32 - Paul Marillonnet

Par exemple, les tests passent du premier coup, sans dossier de MDs préalable, avec la modif suivante :

```
(main) paul@eosandbox:~/eodevel/django-mellon$ git diff
diff --git a/mellon/__init__.py b/mellon/__init__.py
index c9c8f2b..66f34c6 100644
--- a/mellon/__init__.py
+++ b/mellon/__init__.py
@@ -1,6 +1,10 @@
+import os
 try:
     import lasso
```

```
 except ImportError:
     print('***** django-mellon needs the Python binding for the lasso library, *****')
     print('***** look on http://lasso.entrouvert.org/download/ to obtain it *****')
     raise
+
+if not os.path.exists('metadata-cache'):
+    os.makedirs('metadata-cache')
```

#91 - 11 janvier 2018 11:34 - Benjamin Dauvergne

Je ne sais pas ce que c'est que ce dossier metadata-cache normalement tout doit partir dans un sous répertoire de settings.MEDIA_ROOT sinon ça ne va pas.

#92 - 11 janvier 2018 11:37 - Benjamin Dauvergne

Ok j'ai compris c'est n'importe nawak :) J'ai vraiment rien lu, là:

```
federation_utils.py:
 40         unix_path = default_storage.path(path)
 41         if not os.path.exists('metadata-cache'):
 42             os.makedirs('metadata-cache')
 43         f = open(unix_path, 'w')
```

des fois tu passes par default_storage puis ensuite tu fais des accès direct dans le répertoire courant :) faut pas, il faut toujours utilser default_storage qui en plus se débrouille pour créer les répertoires manquant, il est inutile de faire des makedirs avec default_storage.

#93 - 11 janvier 2018 11:43 - Paul Marillonnet

Ok, je viens de vérifier, c'est de ma part une mauvaise compréhension de ton code proposé pour load_federation_cache(), que j'ai étendu de façon erronée en confondant default_storage et os.path. Je corrige mon patch.

(settings.MEDIA_ROOT ? Pas settings.DEFAULT_FILE_STORAGE ?)

#94 - 11 janvier 2018 11:59 - Benjamin Dauvergne

DEFAULT_FILE_STORAGE donne la classe, par défaut c'est FileSystemStorage qui définit l'emplacement comme étant MEDIA_ROOT, tu peux utiliser directement les outils comme os.path.* et file ou open mais uniquement sur des chemins obtenu via default_storage pas sur des chemins que tu construits toi même.

#95 - 11 janvier 2018 12:03 - Paul Marillonnet

Benjamin Dauvergne a écrit :

> Ok j'ai compris c'est n'importe nawak :) J'ai vraiment rien lu, là:
> [...]
>
> DEFAULT_FILE_STORAGE donne la classe, par défaut c'est FileSystemStorage qui définit l'emplacement comme étant MEDIA_ROOT, tu peux utiliser directement les outils comme os.path.* et file ou open mais uniquement sur des chemins obtenu via default_storage pas sur des chemins que tu construits toi même.

Ok, compris.
Je corrige pour qqchose comme :

```
 40         unix_path = default_storage.path(path)
 41         dirname = os.path.dirname(unix_path)
 42         if not os.path.exists(dirname):
 43             os.makedirs(dirname)
 44         f = open(unix_path, 'w')
```

#96 - 11 janvier 2018 12:15 - Benjamin Dauvergne

Oui mais bon il y en a plein d'autres qui ne sont pas bon dans le fichier, j'ai pointé celui-là mais il faut relire toutes les lignes de federation_utils qui font des accès fichier.

#97 - 11 janvier 2018 16:04 - Paul Marillonnet

Je continue de chercher les modifs nécessaires pour avoir le comportement attendu, à savoir plus aucun fichier de métadonnées issues de tests une fois ces tests terminés.

J'ai uniformisé le code mellon de mon patch pour que tous les accès aux fichiers passent par default_storage.

En l'état actuel le patch crée des métadonnées en cache sur le système de fichier, pour tout IdP (que celui-ci provienne d'une fédération ou non).

Et donc, pour faire tourner les tests avec le comportement attendu décrit plus haut, il faudrait paramétrer une fixture tmpdir en tant que MEDIA_ROOT pour chacun des tests unitaires de mellon.

Donc je vois deux options :
- soit je peux faire ça rapidement, sans changer la signature de tous les tests, et alors je commence par modifier mon patch pour que tous les tests passent (+ ceux à ajouter suggérés par Frédéric)
- sinon je commence par modifier le patch pour que la mise en cache ne concerne que les MDs d'IdPs issus d'une fédération, et alors la modification sur MEDIA_ROOT n'est nécessaire que pour une poignée de tests seulement, et je m'occuperai ensuite des corrections de bogue dans mon patch.

Si l'un d'entre vous voit la solution rapidement pour la première option, je suis preneur, sinon il faudrait sûrement que je commence par l'option 2.

#98 - 11 janvier 2018 16:14 - Benjamin Dauvergne

Tu peux faire une fixture "autouse": https://docs.pytest.org/en/latest/fixture.html#autouse-fixtures-xunit-setup-on-steroids

```
@pytest.fixture(autouse=True)
def mellon_settings(settings, tmpdir):
    settings.MEDIA_ROOT = str(tmpdir)
```

#99 - 11 janvier 2018 16:16 - Benjamin Dauvergne

Tu peux aussi polluer MEDIA_ROOT dans django-mellon/testsettings.py pour être sûr que tu n'écris pas n'importe où (tu crées un fichier /tmp/tempfile et tu mets ça comme MEDIA_ROOT, makedirs('/tmp/tempfile/whatever/') va forcément foirer).

#100 - 11 janvier 2018 16:22 - Paul Marillonnet

Benjamin Dauvergne a écrit :

> Tu peux faire une fixture "autouse"

Exactement ce que je cherchais, merci beaucoup, je pars donc sur l'option 1.

#101 - 11 janvier 2018 18:49 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

- Fichier inbetween.diff ajouté

Ci-joint le nouveau patch avec :
- les métadonnées volatiles pour l'exécution des tests
- la création du dossier metadata-cache pour la mise en cache des MDs des IdPs (sans doute à retirer dans un second temps, si on décide ne plus mettre en cache les MDs d'IdPs non fédérés, comme expliqué plus haut par Benjamin).

Benjamin Dauvergne a écrit :

> J'ai appliqué le patch de mon coté tout me semble ok sauf que sur stretch j'ai un souci avec le test test_invalid_msg_on_artifact_resolve mais ça ne semble pas venir de tes modifications, plutôt de la version de Lasso dans stretch, je suis en train de refaire un paquet pour stretch quand ce sera bon je te dis quoi.

Pour moi aussi maintenant tous les tests passent sauf test_invalid_msg_on_artifact_resolve.
Et donc sûrement un souci aussi de mon côté sous Stretch. Est-ce que le paquet à mettre à jour dont tu parles est python-lasso ou liblasso ?

Je joins aussi le git-diff avant le commmit --amend, pour voir directement les dernières modifications par rapport aux précédents patches.

Je vais maintenant écrire les tests unitaires suggérés par Frédéric, et tester des scénarii de SSO avec mellon patché.

#102 - 11 janvier 2018 19:14 - Paul Marillonnet

Frédéric Péters a écrit :

> Plus généralement il y a des passages qui laissent songeur, pourquoi donc charger le fichier pour en extraire un entity id qu'on a déjà :

Oups, pardon, je viens de voir que je n'ai pas inclus ta remarque dans mon dernier patch, je viens de corriger à l'instant.

#103 - 11 février 2018 02:06 - Benjamin Dauvergne

Encore des petits soucis:

```
_____ test_invalid_msg_on_artifact_resolve _____
```

```
private_settings = <django.conf.LazySettings object at 0x7f8629d224d0>, client = <django.test.client.Client ob
ject at 0x7f86235e8850>
caplog = <_pytest.logging.LogCaptureFixture object at 0x7f86235e8d10>, artifact = 'AAQAALQUO+cobSry7mQpUjWDhKk
aePFoeHh4eHh4eHh4eHh4eHh4eHg='

    def test_invalid_msg_on_artifact_resolve(private_settings, client, caplog, artifact):
        private_settings.MELLON_IDENTITY_PROVIDERS = [{
            'METADATA': open('tests/metadata.xml').read(),
        }]
        with HTTMock(html_response):
            client.get('/login/?SAMLart=%s' % artifact)
>       assert 'ArtifactResolveResponse is malformed' in caplog.text
E       AssertionError: assert 'ArtifactResolveResponse is malformed' in 'federation_utils.py        212 WARNI
NG  No JSON settings file for EntityID None\nfederation_utils.py        212 WARNIN...LassoServer object, you m
ust use the methods lasso_server_add_provider() or lasso_server_add_provider_from_buffer().>\n'
E        +  where 'federation_utils.py        212 WARNING  No JSON settings file for EntityID None\nfederation
_utils.py        212 WARNIN...LassoServer object, you must use the methods lasso_server_add_provider() or lass
o_server_add_provider_from_buffer().>\n' = <_pytest.logging.LogCaptureFixture object at 0x7f86235e8d10>.text

tests/test_views.py:248: AssertionError
---------------------------------------------------------------------------- Captured stderr call -------------
-----------------------------------------------------------------
federation_utils.py        212 WARNING  No JSON settings file for EntityID None
federation_utils.py        212 WARNING  No JSON settings file for EntityID 'http://idp5/metadata'
views.py                   309 ERROR    unexpected lasso error
Traceback (most recent call last):
  File "/home/bdauvergne/wd/eo/django-mellon/mellon/views.py", line 276, in continue_sso_artifact
    login.processResponseMsg(result.content)
  File "/tmp/tox-bdauvergne/django-mellon/dj18/local/lib/python2.7/site-packages/lasso.py", line 7313, in proc
essResponseMsg
    Error.raise_on_rc(rc)
  File "/tmp/tox-bdauvergne/django-mellon/dj18/local/lib/python2.7/site-packages/lasso.py", line 62, in raise_
on_rc
    raise exception
ServerProviderNotFoundError: <lasso.ServerProviderNotFoundError(-201): The identifier of a provider is unknown
 to #LassoServer. To register a provider in a #LassoServer object, you must use the methods lasso_server_add_p
rovider() or lasso_server_add_provider_from_buffer().>
---------------------------------------------------------------------------- Captured log call --------------
-----------------------------------------------------------------
federation_utils.py        212 WARNING  No JSON settings file for EntityID None
federation_utils.py        212 WARNING  No JSON settings file for EntityID 'http://idp5/metadata'
views.py                   274 INFO     Got SAML Artifact Response
views.py                   309 ERROR    unexpected lasso error
Traceback (most recent call last):
  File "/home/bdauvergne/wd/eo/django-mellon/mellon/views.py", line 276, in continue_sso_artifact
    login.processResponseMsg(result.content)
  File "/tmp/tox-bdauvergne/django-mellon/dj18/local/lib/python2.7/site-packages/lasso.py", line 7313, in proc
essResponseMsg
    Error.raise_on_rc(rc)
  File "/tmp/tox-bdauvergne/django-mellon/dj18/local/lib/python2.7/site-packages/lasso.py", line 62, in raise_
on_rc
    raise exception
ServerProviderNotFoundError: <lasso.ServerProviderNotFoundError(-201): The identifier of a provider is unknown
 to #LassoServer. To register a provider in a #LassoServer object, you must use the methods lasso_server_add_p
rovider() or lasso_server_add_provider_from_buffer().>
```

#104 - 11 février 2018 02:25 - Benjamin Dauvergne

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

Oublie ma dernière remarque, par contre path avec une correction due à pytest >= 3.4 (les messages de DEBUG n'étaient plus loggé, j'ai ajouté un caplog.set_level(DEBUG)).

#105 - 13 février 2018 08:45 - Paul Marillonnet

Hmm OK merci j'avais pas vu ça.
Si c'est OK pour toi je commence à modifier le patch pour ne mettre en cache que les MDs issues d'une fédération, comme tu me le conseillais plus tôt.

#106 - 13 février 2018 12:22 - Benjamin Dauvergne

Oui.

#107 - 14 février 2018 11:45 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

Paul Marillonnet a écrit :

> ne mettre en cache que les MDs issues d'une fédération, comme tu me le conseillais plus tôt.

Je pense que ce serait quelque chose comme ça. En particulier DefaultAdapter.get_idps.

Je vais maintenant revoir le partage des fonctionnalités prises en charge par get_idps et get_identity_providers_settings (cette dernière méthode ne porte plus vraiment bien son nom...)

#108 - 15 février 2018 10:36 - Paul Marillonnet

Paul Marillonnet a écrit :

> Je vais maintenant revoir le partage des fonctionnalités prises en charge par get_idps et get_identity_providers_settings (cette dernière méthode ne porte plus vraiment bien son nom...)

Au temps pour moi, je pense maintenant qu'il faut lire "Get identity providers from Mellon settings" (et donc le nom n'indique pas qu'il s'agit d'une fonction qui récupère les settings d'un fournisseur d'identité).

#110 - 07 mars 2018 17:59 - Benjamin Dauvergne

- Assigné à changé de Paul Marillonnet à Benjamin Dauvergne

#112 - 12 mars 2018 15:02 - Benjamin Dauvergne

- Version cible mis à 1.2.34

- Patch proposed changé de Oui à Non

#113 - 13 mars 2018 22:55 - Benjamin Dauvergne

- Version cible changé de 1.2.34 à 1.2.35

#114 - 18 avril 2018 11:05 - Paul Marillonnet

- Assigné à changé de Benjamin Dauvergne à Paul Marillonnet

C'est reparti.

#115 - 18 avril 2018 11:28 - Paul Marillonnet

Benjamin Dauvergne a écrit :

> Idéalement on ne devrait charger que la métadonnée du fournisseur qui nous intéresse, et ne plus mettre en cache lasso.Server() (qui ne coûte pas si cher à créer en fait).
>
> L'approche pour cela est celle faite dans authentic2, après un login.processAuthnResponseMsg(), login.processResponseMsg(), logout.processRequestMsg() ou un logout.processResponseMsg, on intercepte les exceptions lasso.ProfileUnknownProviderError ou lasso.ServerProviderNotFoundError, et on tente de charge le fournisseur dont l'entityID est dans login/logout.remoteProviderId, si au deuxième coup ça foire,

Et l'approche que tu décris, pour moi c'est qu'une partie seulement des changements à faire : c'est plutôt la répercussion dans mellon.views de ce chargement "lazy" des métadonnées.

Ça ne sera justifié qu'une fois que mellon.utils.create_server aura été modifiée pour prendre en compte cette minimisation des MDs chargées.

Je me plante ?

#116 - 18 avril 2018 12:14 - Paul Marillonnet

Dans l'idée, c'est quelque chose comme ça que j'aimerais faire fonctionner (diff à partir du précédent patch) :

```
(main2) paul@eosandbox:~/eodevel/django-mellon/mellon$ git diff
diff --git a/mellon/utils.py b/mellon/utils.py
index d505fb4..12a6f64 100644
--- a/mellon/utils.py
+++ b/mellon/utils.py
@@ -76,19 +76,19 @@ def create_server(request):
             password = key[1]
```

```
             key = key[0]
         server.setEncryptionPrivateKeyWithPassword(key, password)
-    for idp in get_idps():
-        try:
-            metadata = idp.get('METADATA')
-            if idp_metadata_is_file(metadata):
-                with open(metadata, 'r') as f:
-                    metadata = f.read()
-            server.addProviderFromBuffer(lasso.PROVIDER_ROLE_IDP, metadata)
-        except lasso.Error, e:
-            logger.error(u'bad metadata in idp %r', idp)
-            logger.debug(u'lasso error: %s', e)
-        except IOError, e:
-            logger.warning('No such metadata file: %r', metadata)
-            continue
+    #for idp in get_idps():
+    #    try:
+    #        metadata = idp.get('METADATA')
+    #        if idp_metadata_is_file(metadata):
+    #            with open(metadata, 'r') as f:
+    #                metadata = f.read()
+    #        server.addProviderFromBuffer(lasso.PROVIDER_ROLE_IDP, metadata)
+    #    except lasso.Error, e:
+    #        logger.error(u'bad metadata in idp %r', idp)
+    #        logger.debug(u'lasso error: %s', e)
+    #    except IOError, e:
+    #        logger.warning('No such metadata file: %r', metadata)
+    #        continue
     return server

diff --git a/mellon/views.py b/mellon/views.py
index d6d58e9..f67ea10 100644
--- a/mellon/views.py
+++ b/mellon/views.py
@@ -120,6 +120,16 @@ class LoginView(ProfileMixin, LogMixin, View):
                 lasso.ProfileStatusNotSuccessError,
                 lasso.ProfileRequestDeniedError):
             self.show_message_status_is_not_success(login, 'SAML authentication failed')
+        except (lasso.ProfileUnknownProviderError,
+                lasso.ServerProviderNotFoundError):
+            try:
+                metadata = utils.get_idp(login.RemoteProviderId)
+                if federation_utils.idp_metadata_is_file(metadata):
+                    with open(metadata, 'r') as f:
+                        metadata = f.read()
+                server.addProviderFromBuffer(lasso.PROVIDER_ROLE_IDP, metadata)
+            except lasso.Error as e:
+                return HttpResponseBadRequest('error processing the authentication response: %r' % e)
         except lasso.Error as e:
             return HttpResponseBadRequest('error processing the authentication response: %r' % e)
         else:
@@ -275,6 +285,17 @@ class LoginView(ProfileMixin, LogMixin, View):
         try:
             login.processResponseMsg(result.content)
             login.acceptSso()
+        except (lasso.ProfileUnknownProviderError,
+                lasso.ServerProviderNotFoundError):
+            try:
+                idp_metadata = utils.get_idp(login.RemoteProviderId)
+                if federation_utils.idp_metadata_is_file(idp_metadata):
+                    with open(metadata, 'r') as f:
+                        metadata = f.read()
+                server.addProviderFromBuffer(lasso.PROVIDER_ROLE_IDP, metadata)
+            except lasso.Error as e:
+                self.log.exception('unexpected lasso error')
+                return HttpResponseBadRequest('error processing the authentication response: %r' % e)
         except lasso.ProfileMissingResponseError:
             # artifact is invalid, idp returned no response
             self.log.warning('ArtifactResolveResponse is empty: dead artifact %r', artifact)
@@ -397,6 +418,16 @@ class LogoutView(ProfileMixin, LogMixin, View):
         self.profile = logout = utils.create_logout(request)
         try:
             logout.processRequestMsg(request.META['QUERY_STRING'])
+        except (lasso.ProfileUnknownProviderError,
+                lasso.ServerProviderNotFoundError):
```

```
+            try:
+                idp_metadata = utils.get_idp(logout.RemoteProviderId)
+                if federation_utils.idp_metadata_is_file(idp_metadata):
+                    with open(metadata, 'r') as f:
+                        metadata = f.read()
+                server.addProviderFromBuffer(lasso.PROVIDER_ROLE_IDP, metadata)
+            except lasso.Error as e:
+                return HttpResponseBadRequest('error processing logout request: %r' % e)
         except lasso.Error as e:
             return HttpResponseBadRequest('error processing logout request: %r' % e)
         try:
@@ -460,6 +491,16 @@ class LogoutView(ProfileMixin, LogMixin, View):
             self.show_message_status_is_not_success(logout, 'SAML logout failed')
         except lasso.LogoutPartialLogoutError:
             self.log.warning('partial logout')
+        except (lasso.ProfileUnknownProviderError,
+                lasso.ServerProviderNotFoundError):
+            try:
+                idp_metadata = utils.get_idp(logout.RemoteProviderId)
+                if federation_utils.idp_metadata_is_file(idp_metadata):
+                    with open(metadata, 'r') as f:
+                        metadata = f.read()
+                server.addProviderFromBuffer(lasso.PROVIDER_ROLE_IDP, metadata)
+            except lasso.Error as e:
+                return HttpResponseBadRequest('error processing a logout request: %s' % e)
         except lasso.Error as e:
             self.log.warning('unable to process a logout response: %s', e)
             return HttpResponseRedirect(resolve_url(settings.LOGIN_REDIRECT_URL))
```

#117 - 18 avril 2018 15:51 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

- Fichier 0002-WIP-federation-file-lazy-provider-loading-19396.patch ajouté

- Patch proposed changé de Non à Oui

Je sépare en deux patches.
Les tests ne passent pas encore, je pose quand même les patches ici et je corrige dès que possible.

Edit: je viendrai aussi modifier les tests pour que le chargement lazy soit correctement testé.

#118 - 19 avril 2018 11:44 - Benjamin Dauvergne

Je note aussi une nouvelle API dans lasso dont tu vas pouvoir te servir:

```
lasso.profileGetIssuer(request.META.get('QUERY_STRING') or request.body)
```

qui te donnera directement l'entityID de la requête entrante sans devoir faire un processRequestMsg.

#119 - 25 avril 2018 14:00 - Paul Marillonnet

C'est noté, merci.

J'ai encore des ProfileMissingRemoteProvideridError et des ServerProviderNotFoundError dont l'origine m'échappe pour l'instant.

Je vais relire le code des vues mellon, que je ne connais pas bien, et je pense que ça m'apparaîtra plus clair ensuite.

#120 - 09 mai 2018 09:25 - Paul Marillonnet

Toujours au même endroit, bien bloqué pour l'instant.
Je continue de chercher l'endroit où le chargement se passe mal.
J'ai essayé de rajouter la double capture d'exceptions partout où on tente d'accéder à l'IdP, mais il doit m'en manquer quelques uns. Je ne vois pas d'autres explications actuellement.

Edit: Je me mange des HTTP 400 lors de l'exécution des tests, et je n'arrive pas encore à voir quelle partie du code capture l'exception et renvoie ce code HTTP. Je continue d'éplucher le code des vues.

Edit2: j'essaie d'ajouter ProfileMissingRemoteProviderError en plus des deux autres classes d'exception capturées lors de la première des deux tentatives de chargement du fournisseur.

#121 - 23 mai 2018 09:59 - Paul Marillonnet

Paul Marillonnet a écrit :

> Edit2: j'essaie d'ajouter ProfileMissingRemoteProviderError en plus des deux autres classes d'exception capturées lors de la première des deux

tentatives de chargement du fournisseur.

Non, ce n'est pas encore ça, les ProfileMissingRemoteProvideridError ne peuvent pas être traitée de la même manière.
Je vais regarder dans le code lasso exactement ce qui dans la phase de login provoque une telle exception.

#122 - 23 mai 2018 16:38 - Paul Marillonnet

Je perds le RelayState dans les paramètres de réponse, bug revelé les tests test_sp_initiated_login et test_sp_initiated_login_chosen.
J'ai creusé mais pas encore compris pourquoi.

```
================================================================================ FAILURES ==========
================================================================================
_____________________________________________________________________ test_sp_initiated_login_chosen
 _____________________________________________________________________

private_settings = <django.conf.LazySettings object at 0x7fc48be98c10>, client = <django.test.client.Client ob
ject at 0x7fc4854348d0>

    def test_sp_initiated_login_chosen(private_settings, client):
        private_settings.MELLON_IDENTITY_PROVIDERS = [{
            'METADATA': open('tests/metadata.xml').read(),
        }]
        qs = urlencode({
            'entityID': 'http://idp5/metadata',
            'next': '/whatever',
        })
        response = client.get('/login/?' + qs)
        assert response.status_code == 302
        params = parse_qs(urlparse(response['Location']).query)
        assert response['Location'].startswith('http://idp5/singleSignOn?')
>       assert set(params.keys()) == set(['SAMLRequest', 'RelayState'])
E       AssertionError: assert set(['SAMLRequest']) == set(['RelayState', 'SAMLRequest'])
E         Extra items in the right set:
E         'RelayState'
E         Use -v to get the full diff

tests/test_views.py:192: AssertionError
_____________________________________________________________________ test_sp_initiated_login __
_____________________________________________________________________

private_settings = <django.conf.LazySettings object at 0x7fc48be98c10>, client = <django.test.client.Client ob
ject at 0x7fc48529acd0>

    def test_sp_initiated_login(private_settings, client):
        private_settings.MELLON_IDENTITY_PROVIDERS = [{
            'METADATA': open('tests/metadata.xml').read(),
        }]
        response = client.get('/login/?next=%2Fwhatever')
        assert response.status_code == 302
        params = parse_qs(urlparse(response['Location']).query)
        assert response['Location'].startswith('http://idp5/singleSignOn?')
>       assert set(params.keys()) == set(['SAMLRequest', 'RelayState'])
E       AssertionError: assert set(['SAMLRequest']) == set(['RelayState', 'SAMLRequest'])
E         Extra items in the right set:
E         'RelayState'
E         Use -v to get the full diff

tests/test_views.py:174: AssertionError
---------- generated xml file: /home/paul/eodevel/django-mellon/jun
it-coverage-dj18-sqlite.xml ----------

---------- coverage: platform linux2, python 2.7.14-final-0 ----------
Coverage XML written to file coverage.xml

================================================================================ 2 failed, 40 passed in 2.18 seco
nds ================================================================================
ERROR: InvocationError: '/tmp/tox-paul/django-mellon/coverage-dj18-sqlite/bin/py.test --random --junit-xml=jun
it-coverage-dj18-sqlite.xml --cov=mellon --cov-report xml tests'
_____________________________________________________________________ summary _____________
_____________________________________________________________________
```

#123 - 24 mai 2018 11:08 - Benjamin Dauvergne

Paul Marillonnet a écrit :

> Je perds le RelayState dans les paramètres de réponse, bug revelé les tests test_sp_initiated_login et test_sp_initiated_login_chosen.
> J'ai creusé mais pas encore compris pourquoi.
>
> [...]

Use the

```
import pdb
pdb.set_trace()
```

luke !

#124 - 31 mai 2018 17:44 - Paul Marillonnet

- Fichier 0001-federation-file-loading-19396.patch ajouté

phew

#125 - 31 mai 2018 17:59 - Frédéric Péters

(il y a toute une partie "réessais" qui ne m'a pas l'air d'être liée à la possibilité d'utiliser un fichier de fédération) (je n'ai pas relu tout l'historique mais de rapides recherches rien) (cette partie du patch n'existait pas dans le précédent).

#126 - 31 mai 2018 18:07 - Paul Marillonnet

Oui, pour ne pas avoir à charger tout le fichier à chaque login/out. C'était la dernière étape du ticket, mais ça n'a peut-être pas sa place ici.
Un patch à part ?
Carrément un ticket à part ?

#127 - 31 mai 2018 18:11 - Frédéric Péters

> Oui, pour ne pas avoir à charger tout le fichier à chaque login/out. C'était la dernière étape du ticket, mais ça n'a peut-être pas sa place ici.

(Je vais laisser la relecture à d'autres.)

#128 - 31 mai 2018 18:15 - Paul Marillonnet

Frédéric Péters a écrit :

> (Je vais laisser la relecture à d'autres.)

(Si tu lâches l'affaire parce que tu trouves que c'est n'imp' ce patch, je suis quand même preneur de conseils dans les grandes lignes pour corriger le tir si besoin est. Ça facilitera le travail des relecteurs de toute façon.)

#129 - 31 mai 2018 18:28 - Frédéric Péters

En soit, juste regarder le diffstat genre 600 lignes qui bougent sur les .py (sur un projet qui en fait 1175), c'est signe que vraiment beaucoup de choses arrivent ou vont et viennent, et ça rend la tâche de relecture bien trop pesante. (à titre de compaison, en recherche rapide, côté w.c.s. pour un patch qui bouge tant, en valeur absolue, il faut remonter à décembre 2015 (actions globales)).

#130 - 01 juin 2018 16:58 - Paul Marillonnet

Ok, je vais rebaser (oublié de le faire dans mon précédent patch) et découper le patch en trois :
0001 - le chargement de fichier de fédération
0002 - les réessais sur les login/out dans les vues
0003 - les tests

#131 - 01 juin 2018 17:06 - Benjamin Dauvergne

Paul Marillonnet a écrit :

> Ok, je vais rebaser (oublié de le faire dans mon précédent patch) et découper le patch en trois :
> 0001 - le chargement de fichier de fédération
> 0002 - les réessais sur les login/out dans les vues
> 0003 - les tests

Un truc que je disais plus haut pour ne pas faire de réessai mais que tu as du rater:

Je note aussi une nouvelle API dans lasso dont tu vas pouvoir te servir:

```
> lasso.profileGetIssuer(request.META.get('QUERY_STRING') or request.body)
>
```

qui te donnera directement l'entityID de la requête entrante sans devoir faire un processRequestMsg.

#132 - 01 juin 2018 17:43 - Paul Marillonnet

Ah oui, j'ai laissé passer ça.
Je ne comprends pas pour l'instant en quoi l'obtention directe de l'EntityID permet de s'épargner les réessais lorsque que le fournisseur est inconnu du serveur lasso.
Je vais lire le code de cette nouvelle interface.

#133 - 01 juin 2018 17:52 - Frédéric Péters

```
provider_id = lasso.profileGetIssuer(...)
if unknown provider_id:
    load_federations_until(provider_id)
...
login.process...
```

#134 - 13 juin 2018 15:08 - Paul Marillonnet

Oui OK je comprends, merci.
La montée en version de liblasso3 et de python-lasso casse l'exécution correcte de certains tests sur mon installation.
Je regarde ce qui ne va pas.

#135 - 21 juin 2018 11:28 - Paul Marillonnet

J'ai rebasé sur master et inclus les modifs induites par #24531.
Je travaille sur le module federation_utils du patch précédent, qui n'est pas interprétable par python3 pour l'instant.
Une fois que c'est fait, j'inclus la nouvelle API dans les vues pour s'épargner les rejeux de chargement dans mellon.

#136 - 27 juin 2018 10:50 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

- Patch proposed changé de Oui à Non

Je pose ici, pour info (et si jamais mon ordi à moi aussi plante :)), le code interprétable par python 2 et 3. Je passe à l'inclusion de la nouvelle API.

#137 - 27 juin 2018 14:35 - Paul Marillonnet

J'ai des InvocationError encore un peu mystérieuses pour moi au moment de l'appel à la nouvelle API. Même avec le plus petit bout de code possible, à la place du rejeu des vues :

```
        payload = request.META.get('QUERY_STRING') or request.body
        provider_id = lasso.profileGetIssuer(payload.decode('utf-8'))
        server = utils.recreate_server(request, provider_id)
```

Et des erreurs très peu verbeuses :

```
GLOB sdist-make: /home/paul/src/django-mellon/setup.py
coverage-py2-dj18-pg develop-inst-nodeps: /home/paul/src/django-mellon
coverage-py2-dj18-pg installed: atomicwrites==1.1.5,attrs==18.1.0,beautifulsoup4==4.6.0,certifi==2018.4.16,cha
rdet==3.0.4,coverage==4.5.1,cssselect==1.0.3,Django==1.8.19,-e git://repos.entrouvert.org/django-mellon.git@dc
caab137d8c479c8f94db5fff12d999d6a028ef#egg=django_mellon,django-webtest==1.9.2,funcsigs==1.0.2,httmock==1.2.6,
idna==2.7,isodate==0.6.0,lxml==4.2.2,mock==2.0.0,more-itertools==4.2.0,pbr==4.0.4,pluggy==0.6.0,psycopg2==2.7.
5,py==1.5.3,pyquery==1.4.0,pytest==3.6.2,pytest-cov==2.5.1,pytest-django==3.3.2,pytest-mock==1.10.0,pytest-ran
dom==0.2,pytz==2018.4,requests==2.19.1,six==1.11.0,urllib3==1.23,waitress==1.1.0,WebOb==1.8.2,WebTest==2.0.30
coverage-py2-dj18-pg runtests: PYTHONHASHSEED='2221710909'
coverage-py2-dj18-pg runtests: commands[0] | ./getlasso.sh
'/tmp/tox-paul/django-mellon/coverage-py2-dj18-pg/lib/python2.7/site-packages/lasso.py' -> '/usr/lib/python2.7
/dist-packages/lasso.py'
'/tmp/tox-paul/django-mellon/coverage-py2-dj18-pg/lib/python2.7/site-packages/_lasso.x86_64-linux-gnu.so' -> '
/usr/lib/python2.7/dist-packages/_lasso.x86_64-linux-gnu.so'
coverage-py2-dj18-pg runtests: commands[1] | py.test --random --junit-xml=junit-coverage-py2-dj18-pg.xml --cov
=mellon --cov-report xml tests
============================================================ test session starts ============================
================================
```

```
platform linux2 -- Python 2.7.15, pytest-3.6.2, py-1.5.3, pluggy-0.6.0
Tests are shuffled using seed number 391706339936.
Django settings: testsettings (from environment variable)
rootdir: /home/paul/src/django-mellon, inifile:
plugins: random-0.2, mock-1.10.0, django-3.3.2, cov-2.5.1, django-webtest-1.9.2
collected 43 items


tests/test_utils.py .
                        [  2%]
tests/test_default_adapter.py .
                        [  4%]
tests/test_sso_slo.py ERROR: InvocationError for command '/tmp/tox-paul/django-mellon/coverage-py2-dj18-pg/bin
/py.test --random --junit-xml=junit-coverage-py2-dj18-pg.xml --cov=mellon --cov-report xml tests' (exited with
 code -11)
```

Je cherche.

#138 - 27 juin 2018 17:51 - Paul Marillonnet

L'InvocationError de tox semble résulter d'une erreur de segmentation à l'exécution de la fonction profile_get_issuer de la lib lasso.
Benj, est-ce que tu saurais me dire si cette nouvelle fonction introduite dans la version 2.6 de la lib est couverte par des tests unitaires que je pourrais faire tourner sur ma machine ?
Je voudrais essayer une exécution en dehors de mellon pour voir si le problème vient de là ou non.

Edit: je vais essayer de faire tourner lasso/tests/basic_tests.c

#139 - 27 juin 2018 18:05 - Frédéric Péters

> Je voudrais essayer une exécution en dehors de mellon pour voir si le problème vient de là ou non.

```
>>> msg = 'SAMLRequest=fZHbasMwDIZfJfi+sXtYk5okkC4NFLZRtrGL3QzXUWnAsTNL2eHt56YUOga7Ekj69P+SMlSd6WU50NE+wvsASNF
XZyzKsZCzwVvpFLYoreoAJWn5VN7fyVksZO8dOe0Mu0L+JxQieGqdZdG2ytmbSNJkvZiv01WdJDdJLRbLzXyTlnVZ1etVmrLoBTyG/pwFPECIA
2wtkrIUUmKaTsRyMkuep0spFlLMXllUhR1aq2ikjkQ9Ss5VWBAstTo+eGhi47QysWjiPfC26fnJ+YwjOhbVzmsY75GzgzIIJ9VdMN5+wCVTZCd
Ajm58cdHQrtu7v/OV1m6whLwDY5wNgVSjSPGMX0/Jzp94CDfbVjtnWv0dlQH4vPWgKEiTH4Dx4kz9flnxAw==&SigAlg=http://www.w3.org
/2000/09/xmldsig'
>>> import lasso
>>> lasso.profileGetIssuer(msg)
'https://combo.fred.local.0d.be/accounts/mellon/metadata/'
```

#140 - 27 juin 2018 18:07 - Frédéric Péters

Après ça plante facilement quand ça rencontre n'importe quoi, ce qui est un bug en soit,

```
>>> msg = 'SAMLRequest=Hello!!!fZHbasMwDIZfJfi+sXtYk5okkC4NFLZRtrGL3QzXUWnAsTNL2eHt56YUOga7Ekj69P+SMlSd6WU50NE
+wvsASNFXZyzKsZCzwVvpFLYoreoAJWn5VN7fyVksZO8dOe0Mu0L+JxQieGqdZdG2ytmbSNJkvZiv01WdJDdJLRbLzXyTlnVZ1etVmrLoBTyG/
pwFPECIA2wtkrIUUmKaTsRyMkuep0spFlLMXllUhR1aq2ikjkQ9Ss5VWBAstTo+eGhi47QysWjiPfC26fnJ+YwjOhbVzmsY75GzgzIIJ9VdMN5
+wCVTZCdAjm58cdHQrtu7v/OV1m6whLwDY5wNgVSjSPGMX0/Jzp94CDfbVjtnWv0dlQH4vPWgKEiTH4Dx4kz9flnxAw==&SigAlg=http://ww
w.w3.org/2000/09/xmldsig'
>>> lasso.profileGetIssuer(msg)
munmap_chunk(): invalid pointer

Aborted (core dumped)
```

(Hello!!! ajouté au début)

Bref, tu rencontres un bug dans lasso, mais parce que tu lui passes quelque chose d'incorrect.

Ton payload.decode('utf-8'), tu peux l'afficher, le copier/coller, etc. ?

#141 - 27 juin 2018 18:18 - Paul Marillonnet

Ah oui, j'aurais quand même cru à autre chose qu'un segfault.

```
coverage-py2-dj111-pg develop-inst-nodeps: /home/paul/src/django-mellon
coverage-py2-dj111-pg installed: atomicwrites==1.1.5,attrs==18.1.0,beautifulsoup4==4.6.0,certifi==2018.4.16,ch
ardet==3.0.4,coverage==4.5.1,cssselect==1.0.3,Django==1.11.13,-e git://repos.entrouvert.org/django-mellon.git@
dccaab137d8c479c8f94db5fff12d999d6a028ef#egg=django_mellon,django-webtest==1.9.2,funcsigs==1.0.2,httmock==1.2.
6,idna==2.7,isodate==0.6.0,lxml==4.2.2,mock==2.0.0,more-itertools==4.2.0,pbr==4.0.4,pluggy==0.6.0,psycopg2==2.
7.5,py==1.5.4,pyquery==1.4.0,pytest==3.6.2,pytest-cov==2.5.1,pytest-django==3.3.2,pytest-mock==1.10.0,pytest-r
andom==0.2,pytz==2018.4,requests==2.19.1,six==1.11.0,urllib3==1.23,waitress==1.1.0,WebOb==1.8.2,WebTest==2.0.3
0
```

```
coverage-py2-dj111-pg runtests: PYTHONHASHSEED='1694759793'
coverage-py2-dj111-pg runtests: commands[0] | ./getlasso.sh
'/tmp/tox-paul/django-mellon/coverage-py2-dj111-pg/lib/python2.7/site-packages/lasso.py' -> '/usr/lib/python2.
7/dist-packages/lasso.py'
'/tmp/tox-paul/django-mellon/coverage-py2-dj111-pg/lib/python2.7/site-packages/_lasso.x86_64-linux-gnu.so' ->
'/usr/lib/python2.7/dist-packages/_lasso.x86_64-linux-gnu.so'
coverage-py2-dj111-pg runtests: commands[1] | py.test --random --junit-xml=junit-coverage-py2-dj111-pg.xml --c
ov=mellon --cov-report xml -x
============================================================================== test session starts ====
============================================================================
platform linux2 -- Python 2.7.15, pytest-3.6.2, py-1.5.4, pluggy-0.6.0
Tests are shuffled using seed number 391709750361.
Django settings: testsettings (from environment variable)
rootdir: /home/paul/src/django-mellon, inifile:
plugins: random-0.2, mock-1.10.0, django-3.3.2, cov-2.5.1, django-webtest-1.9.2
collected 43 items


tests/test_default_adapter.py ...
                                                                      [  6%]
tests/test_sso_slo.py
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> PDB set_trace (IO-capturing turned
 off) >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
> /home/paul/src/django-mellon/mellon/views.py(120)post()
-> provider_id = lasso.profileGetIssuer(payload.decode('utf-8'))
(Pdb) payload.decode('utf-8')
u'RelayState=None&SAMLResponse=PHNhbWxwOlJlc3BvbnNlIHhtbG5zOnNhbWxwPSJ1cm46b2FzaXM6bmFtZXM6dGM6U0FNTDoyLjA6cHJ
vdG9jb2wiIHhtbG5zOnNhbWw9InVybjpvYXNpczpuYW1lczp0YzpTQU1MOjIuMDphc3NlcnRpb24iIElEPSJfMUU2MTg4QjBCNDFBNjdCQjBFN
TQ0RDk0NjEwREVFQzIiIEluUmVzcG9uc2VUbz0iXzBFRDY2RkJCNkNENjU1ODc4NjlGOTMyMDA1NTA4NDNFIiBWZXJzaW9uPSIyLjAiIElzc3V
lSW5zdGFudD0iMjAxOC0wNi0yN1QxNjoxNjo1NVoiIERlc3RpbmF0aW9uPSJodHRwOi8vdGVzdHNlcnZlci9sb2dpbi8iPjxzYW1sOklzc3Vlc
j5odHRwOi8vaWRwNS9tZXRhZGF0YTwvc2FtbDpJc3N1ZXI%2BPFNpZ25hdHVyZSB4bWxucz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94b
Wxkc2lnIyI%2BCjxTaWduZWRJbmZvPgo8Q2Fub25pY2FsaXphdGlvbk1ldGhvZCBBbGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvM
TAveG1sLWV4Yy1jMTRuIyIvPgo8U2lnbmF0dXJlTWV0aG9kIEFsZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnI3J
zYS1zaGExIi8%2BCjxSZWZlcmVuY2UgVVJJPSIjXzFFNjE4OEIwQjQxQTY3QkIwRTU0NEQ5NDYxMERFRUMyIj4KPFRyYW5zZm9ybXM%2BCjxUc
mFuc2Zvcm0gQWxnb3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAwLzA5L3htbGRzaWcjZW52ZWxvcGVkLXNpZ25hdHVyZSIvPgo8VHJhbnN
mb3JtIEFsZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMS8xMC94bWwtZXhjLWMxNG4jIi8%2BCjwvVHJhbnNmb3Jtcz4KPERpZ2VzdE1
ldGhvZCBBbGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvMDkveG1sZHNpZyNzaGExIi8%2BCjxEaWdlc3RWYWx1ZT55eTFHQUpsc1l
6VHRCNElBZm9PN2dwVTRIc3c9PC9EaWdlc3RWYWx1ZT4KPC9SZWZlcmVuY2U%2BCjwvU2lnbmVkSW5mbz4KPFNpZ25hdHVyZVZhbHVlPkdER3N
Nd0xGWjVDVjJONDU3bXVhNmRoWldaNSt0WTh6Zm1mN2hPR2xUNFdyS21sTWYrL041U0IvbS90Mm80VVAKdVdKTHk3N0xycDFrNy85L2EvY2cwN
XYxQXUwL2xjVjFFcy9PU1dxQlk1c2ZMY0pjMUY2a3VBZUdFcmJWY0ZDSAo0bUo2NFFqTURCdTNCMVFOS2EwV1lyeXVGUXhDTVRvT3g2RkpMcGZ
RQVFHZTA2SG5tenhyb3AzcXRGR041KzVGCnNFbXp0T2NIVlBaWERIV2F2cFFraUNYajFZSStkUjhCSWx4eHgwdHluYTQyeE53bW11cTdnWlVZY
2drcWN3U2sKNnZxK3ZFT2NUMEZ5czRvYVhDYktuWTBnUXg5TFl2SlAyaWhmUFBocDU1cmdSOVJiSlkwYlpEbEpGWkRoLzJiUwpaRkFMelBkWGV
lNUJnWnc5elRxN3dBPT08L1NpZ25hdHVyZVZhbHVlPgo8S2V5SW5mbz4KPEtleVZhbHVlPgo8UlNBS2V5VmFsdWU%2BCjxNb2R1bHVzPgp6VG9
mSHBXQWRoSDNCUi8rMWxWVk5HUlZZMnFIM0g0KzhjRGFvZmc1Z3k2b2F6Z0IvcVZUWml4bStldVpGMXdWCmEvVDVTUjBDQmVGRjRKWUJtQzBIV
2wzOWIyYnFvTkdWMElMTEt5akRyRTg4cEhQK2s1UEJGZWI5OHpSQVk5NWYKUERPUGZnRmM0ZzY0Vzc2ZnZyaThxZlh4MzY2NVVBVE9UWG52cW5
GT25pbEEvTWw5MDB1c3Q1RHkvSUt5R2dWVAo0eGdtMm5WUUQ2SFltZzdSanlnYS9MQnRURWVLZ2MzaysrZk01dDhBemhkb05DaUdaL0V6MVJ6d
GFuakVvQnpXCmRTcm1IQUdzZW1NVXhGTFBwUUo4eWdsSVlpTDdmRWt5UTBLTXZSY1REazBwVnptTkVxVE5LUTNtUHdwTXorVFcKTTgrd01jOUZ
qTnRaYUdjMjEzb21XUT09CjwvTW9kdWx1cz4KPEV4cG9uZW50PgpBUUFCCjwvRXhwb25lbnQ%2BCjwvUlNBS2V5VmFsdWU%2BCjwvS2V5VmFsd
WU%2BCjwvS2V5SW5mbz4KPC9TaWduYXR1cmU%2BPHNhbWxwOlN0YXR1cz48c2FtbHA6U3RhdHVzQ29kZSBWYWx1ZT0idXJuOm9hc2lzOm5hbWV
zOnRjOlNBTUw6Mi4wOnN0YXR1czpTdWNjZXNzIi8%2BPC9zYW1scDpTdGF0dXM%2BPHNhbWw6QXNzZXJ0aW9uIFZlcnNpb249IjIuMCIgSUQ9I
l9BQThCQzE3NUMyMjNBRDQyREM5QzE4QkFFODEzMjg4NSIgSXNzdWVJbnN0YW50PSIyMDE4LTA2LTI3VDE2OjE2OjU1WiI%2BPHNhbWw6SXNzd
WVyPmh0dHA6Ly9pZHA1L21ldGFkYXRhPC9zYW1sOklzc3Vlc48U2lnbmF0dXM%2BPHNhbWw6QXNzZXJ0aW9uIFZlcnNpb249IjIuMCIgSUQ9I
tbGRzaWcjIj4KPFNpZ25lZEluZm8%2BCjxDYW5vbmljYWxpemF0aW9uTWV0aG9kIEFsZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMS8
xMC94bWwtZXhjLWMxNG4jIi8%2BCjxTaWduYXR1cmVNZXRob2QgQWxnb3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAwLzA5L3htbGRzaWc
jcnNhLXNoYTEiLz4KPFJlZmVyZW5jZSBVUkk9IiNfQUE4QkMxNzVDMjIzQUQ0MkRDOUMxOEJBRTgxMzI4ODUiPgo8VHJhbnNmb3Jtcz4KPFRyY
W5zZm9ybSBBbGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvMDkveG1sZHNpZyNlbnZlbG9wZWQtc2lnbmF0dXJlIi8%2BCjxUcmFuc
2Zvcm0gQWxnb3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzEwL3htbC1leGMtYzE0biMiLz4KPC9UcmFuc2Zvcm1zPgo8RGlnZXN0TWV
0aG9kIEFsZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnI3NoYTEiLz4KPERpZ2VzdFZhbHVlPjRkSHlYMXkxYzVvM
zNyTDBUNXhZbUc2TGhTWT08L0RpZ2VzdFZhbHVlPgo8L1JlZmVyZW5jZT4KPC9TaWduZWRJbmZvPgo8U2lnbmF0dXJlVmFsdWU%2BcWxheVBhd
ThONUlmZnZGVHBabXYvaHNORG5hdGlEL2lmcGEySmNCUyt5b1lwVE5pU3ZLMlBVVmpuZjV5NmtRdQpxSmZ4QjhoOTVNRG5zclRTQXRvMlI5aTZ
JanNLRThBclZYTjQwLzBldEhyODlzRjlzWnFRYUZ6NW5MVGdTSGN0Cm9SUy8wb0NkaXFxSzI2QndKaE0vZG1rdG40Q2xzSzNIbU5KbWNnMkRVY
kxtTHFOa2dGMDVCMkFQSHVFMVFidkQKc1JST3p3QmJrTmh2VnZHUkVTYTZ3MTlZdnZwQkNXSnhsc0kxUW5VTXAvUktTUDBOcG5TekdvZWIraEh
rUGhmMgpVZEpjNzVWWDh5akpKSUNkaFc0OWtxK09neUQyZnpKTkNURXZKcjZtL2RndjNLYTVjUVpXTHROVFVxU09HMUJ0CjVlMjNndVhNMlNsO
WZ1V1hCa2k3dlE9PTwvU2lnbmF0dXJlVmFsdWU%2BCjxLZXlJbmZvPgo8S2V5VmFsdWU%2BCjxSU0FLZXlWYWx1ZT4KPE1vZHVsdXM%2BCnpUb
2ZIcFdBZGhIM0JSLysxbFZWTkdSVlkycUgzSDQrOGNEYW9mZzVneTZvYXpnQi9xVlRaaXhtK2V1WkYxd1YKYS9UNVNSMENCZUZGNEpZQm1DMEh
XbDM5YjJicW9OR1YwSUxMS3lqRHJFODhwSFArazVQQkZlYjk4elJBWTk1ZgpQRE9QZmdGYzRnNjRXNzZmdnJpOHFmWHgzNjY1VUFUT1RYbnZxb
kZPbmlsQS9NbDkwMHVzdDVEeS9JS3lHZ1ZUCjR4Z20yblZRRDZIWW1nN1JqeWdhL0xCdFRFZUtnYzNrKytmTTV0OEF6aGRvTkNpR1ovRXoxUnp
0YW5qRW9CelcKZFNybUhBR3NlbU1VeEZMUHBRSjh5Z2xJWWlMN2ZFa3lRMEtNdlJjVERrMHBWem1ORXFUTktRM21Qd3BNeitUVwpNOCt3TWM5R
mpOdFphR2MyMTNvbVdRPT0KPC9Nb2R1bHVzPgo8RXhwb25lbnQ%2BCkFRQUIKPC9FeHBvbmVudD4KPC9SU0FLZXlWYWx1ZT4KPC9LZXlWYWx1Z
T4KPC9LZXlJbmZvPgo8L1NpZ25hdHVyZT48c2FtbDpTdWJqZWN0PjxzYW1sOk5hbWVJRCBGb3JtYXQ9InVybjpvYXNpczpuYW1lczp0YzpTQU1
MOjIuMDpuYW1laWQtZm9ybWF0OnBlcnNpc3RlbnQiIE5hbWVRdWFsaWZpZXI9Imh0dHA6Ly9pZHA1L21ldGFkYXRhIiBTUE5hbWVRdWFsaWZpZ
XI9Imh0dHA6Ly90ZXN0c2VydmVyL21ldGFkYXRhLyI%2BXzZFN0E2RDQwNUI3ODQyQjVFOUQ0NDQ3MDk3NzE0MDhBPC9zYW1sOk5hbWVJRD48c
2FtbDpTdWJqZWN0Q29uZmlybWF0aW9uIE1ldGhvZD0idXJuOm9hc2lzOm5hbWVzOnRjOlNBTUw6Mi4wOmNtOmJlYXJlciI%2BPHNhbWw6U3Via
mVjdENvbmZpcm1hdGlvbkRhdGEgTm90T25PckFmdGVyPSJGSVhNRSIgUmVjaXBpZW50PSJodHRwOi8vdGVzdHNlcnZlci9sb2dpbi8iIEluUmV
zcG9uc2VUbz0iXzBFRDY2RkJCNkNENjU1ODc4NjlGOTMyMDA1NTA4NDNFIi8%2BPC9zYW1sOlN1YmplY3RDb25maXJtYXRpb24%2BPC9zYW1sO
```

```
lN1YmplY3Q%2BPHNhbWw6Q29uZGl0aW9ucyBOb3RCZWZvcmU9IkZJWE1FIiBOb3RPbk9yQWZ0ZXI9IkZJWE1FIj48c2FtbDpBdWRpZW5jZVJlc
3RyaWN0aW9uPjxzYW1sOkF1ZGllbmNlPmh0dHA6Ly90ZXN0c2VydmVyL21ldGFkYXRhLzwvc2FtbDpBdWRpZW5jZT48L3NhbWw6QXVkaWVuY2V
SZXN0cmljdGlvbj48L3NhbWw6Q29uZGl0aW9ucz48c2FtbDpBdXRoblN0YXRlbWVudCBBdXRobkluc3RhbnQ9IkZJWE1FIiBTZXNzaW9uSW5kZ
Xg9Il9BQThCQzE3NUMyMjNBRDQyREM5QzE4QkFFODEzMjg4NSI%2BPHNhbWw6QXV0aG5Db250ZXh0PjxzYW1sOkF1dGhuQ29udGV4dENsYXNzU
mVmPnVybjpvYXNpczpuYW1lczp0YzpTQU1MOjEuMDphbTpwYXNzd29yZDwvc2FtbDpBdXRobkNvbnRleHRDbGFzc1JlZj48L3NhbWw6QXV0aG5
Db250ZXh0Pjwvc2FtbDpBdXRoblN0YXRlbWVudD48L3NhbWw6QXNzZXJ0aW9uPjwvc2FtbHA6UmVzcG9uc2U%2B'
(Pdb) c
ERROR: InvocationError for command '/tmp/tox-paul/django-mellon/coverage-py2-dj111-pg/bin/py.test --random --j
unit-xml=junit-coverage-py2-dj111-pg.xml --cov=mellon --cov-report xml -x' (exited with code -11)
```

#142 - 28 juin 2018 11:31 - Paul Marillonnet

Et bien du coup c'est la chaîne passée à cette fonction qui est incorrecte.

```
        payload = request.META.get('QUERY_STRING') or request.body
        provider_id = lasso.profileGetIssuer(payload.decode('utf-8'))
```

On dirait que profileGetIssuer n'aime pas la SAMLResponse chiffrée, ou bien le fait que la totalité de la query string soit passée en argument.

#143 - 28 juin 2018 12:09 - Frédéric Péters

> On dirait que profileGetIssuer n'aime pas la SAMLResponse chiffrée, ou bien le fait que la totalité de la query string soit passée en argument.

Ce qui peut très rapidement se vérifier en ne passant que SAMLResponse=... (réponse : pas ça).

Mais le contenu de SAMLResponse n'arive pas à être décompressé :

```
>>> lasso.Profile.getIssuer(msg.split('&')[1])
2018-06-28 12:03:55,097 - Lasso - ERROR - 2018-06-28 12:03:55 (tools.c/:1373) Failed to inflate
```

(le message passe correctement sur http://rnd.feide.no/simplesaml/module.php/saml2debug/debug.php)

#144 - 28 juin 2018 12:25 - Paul Marillonnet

Oui, en effet, l'appel
lasso_profile_get_issuer -> lasso_xmltextreader_from_message -> lasso_get_saml_message -> lasso_inflate -> zlib::inflate
se passe mal.

#145 - 28 juin 2018 15:58 - Paul Marillonnet

Et donc en lien avec #24853.

#146 - 11 juillet 2018 17:10 - Paul Marillonnet

Et donc encore un comportement étrange, peut-être ai-je mal compris l'API :/

Je suis sûr liblasso3 2.6.0.7.gf33d-1~eob90+1.

J'ai mis un point d'arrêt dans le code, juste avant un process*Msg, pour tester cette API, et le profileGetIssuer ne renvoie rien :

```
paul@DebianStable:~/src/django-mellon$ tox -e coverage-py2-dj18-sqlite
coverage-py2-dj18-sqlite develop-inst-nodeps: /home/paul/src/django-mellon
coverage-py2-dj18-sqlite installed: atomicwrites==1.1.5,attrs==18.1.0,beautifulsoup4==4.6.0,certifi==2018.4.16
,chardet==3.0.4,coverage==4.5.1,cssselect==1.0.3,Django==1.8.19,-e git://repos.entrouvert.org/django-mellon.gi
t@ab0833cc0d3223b4c0648e8698a46c0b2d644272#egg=django_mellon,django-webtest==1.9.2,funcsigs==1.0.2,httmock==1.
2.6,idna==2.7,isodate==0.6.0,lxml==4.2.3,mock==2.0.0,more-itertools==4.2.0,pbr==4.1.0,pkg-resources==0.0.0,plu
ggy==0.6.0,py==1.5.4,pyquery==1.4.0,pytest==3.6.3,pytest-cov==2.5.1,pytest-django==3.3.2,pytest-mock==1.10.0,p
ytest-random==0.2,pytz==2018.5,requests==2.19.1,six==1.11.0,urllib3==1.23,waitress==1.1.0,WebOb==1.8.2,WebTest
==2.0.30
coverage-py2-dj18-sqlite runtests: PYTHONHASHSEED='1015840233'
coverage-py2-dj18-sqlite runtests: commands[0] | ./getlasso.sh
'/tmp/tox-paul/django-mellon/coverage-py2-dj18-sqlite/lib/python2.7/site-packages/lasso.py' -> '/usr/lib/pytho
n2.7/dist-packages/lasso.py'
'/tmp/tox-paul/django-mellon/coverage-py2-dj18-sqlite/lib/python2.7/site-packages/_lasso.x86_64-linux-gnu.so'
-> '/usr/lib/python2.7/dist-packages/_lasso.x86_64-linux-gnu.so'
coverage-py2-dj18-sqlite runtests: commands[1] | py.test --random --junit-xml=junit-coverage-py2-dj18-sqlite.x
ml --cov=mellon --cov-report xml tests
============================================ test session starts ============================================
platform linux2 -- Python 2.7.13, pytest-3.6.3, py-1.5.4, pluggy-0.6.0
Tests are shuffled using seed number 392018000751.
Django settings: testsettings (from environment variable)
rootdir: /home/paul/src/django-mellon, inifile:
```

```
plugins: random-0.2, mock-1.10.0, django-3.3.2, cov-2.5.1, django-webtest-1.9.2
collected 43 items

tests/test_utils.py .                                                                [  2%]
tests/test_default_adapter.py .                                                      [  4%]
tests/test_views.py ...                                                              [ 11%]
tests/test_sso_slo.py
>>>>>>>>>>>>>>>>>>>>>>>>> PDB set_trace (IO-capturing turned off) >>>>>>>>>>>>>>>>>>>>>>>>>
> /home/paul/src/django-mellon/mellon/views.py(123)post()
-> login.processAuthnResponseMsg(request.POST['SAMLResponse'])
(Pdb) payload = request.META.get('QUERY_STRING') or request.body
(Pdb) payload
'SAMLResponse=PHNhbWxwOlJlc3BvbnNlIHhtbG5zOnNhbWxwPSJ1cm46b2FzaXM6bmFtZXM6dGM6U0FNTDoyLjA6cHJvdG9jb2wiIHhtbG5z
OnNhbWw9InVybjpvYXNpczpuYW1lczp0YzpTQU1MOjIuMDphc3NlcnRpb24iIElEPSJfNDFGQkNBREUwREVEOUUzRDgzREYwMzI2MEZGMDVEMD
YiIEluUmVzcG9uc2VUbz0iXzdBNzY1OEI0MkIwRjc5QUM5OTNGMjlBMDRBRjA5NUZEIiBWZXJzaW9uPSIyLjAiIElzc3VlSW5zdGFudD0iMjAx
OC0wNy0xMVQxNDo0NToxN1oiIERlc3RpbmF0aW9uPSJodHRwOi8vdGVzdHNlcnZlci9sb2dpbi8iPjxzYW1sOklzc3Vlcj5odHRwOi8vaWRwNS
9tZXRhZGF0YTwvc2FtbDpJc3N1ZXI%2BPFNpZ25hdHVyZSB4bWxucz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnIyI%2BCjxT
aWduZWRJbmZvPgo8Q2Fub25pY2FsaXphdGlvbk1ldGhvZCBBbGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvMTAveG1sLWV4Yy1jMT
RuIyIvPgo8U2lnbmF0dXJlTWV0aG9kIEFsZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnI3JzYS1zaGExIi8%2BCj
xSZWZlcmVuY2UgVVJJPSIjXzQxRkJDQURFMERFRDlFM0Q4M0RGMDMyNjBGRjA1RDA2Ij4KPFRyYW5zZm9ybXM%2BCjxUcmFuc2Zvcm0gQWxnb3
JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAwLzA5L3htbGRzaWcjZW52ZWxvcGVkLXNpZ25hdHVyZSIvPgo8VHJhbnNmb3JtIEFsZ29yaXRo
bT0iaHR0cDovL3d3dy53My5vcmcvMjAwMS8xMC94bWwtZXhjLWMxNG4jIi8%2BCjwvVHJhbnNmb3Jtcz4KPERpZ2VzdE1ldGhvZCBBbGdvcml0
aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvMDkveG1sZHNpZyNzaGExIi8%2BCjwvVHJhbnNmb3Jtcz4KPERpZ2VzdE1ldGhvZCBBbGdvcml0
aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvMDkveG1sZHNpZyNzYWx25PNmF5ZTMvcWFWZ3FwQlNhK3I2S01h
Z3Jld3c9PC9EaWdlc3RWYWx1ZT4KPC9SZWZlcmVuY2U%2BCjwvU2lnbmVkSW5mbz4KPFNpZ25hdHVyZVZhbHVlPmRweFdZMXp0RUlveEpoRHo5
SkRxSlRpT21hT3JicWJaUXRuNVlRN1N0dEY2a0NrMUYyMFRnKzI5TDMzcEk1ZisKVGdQeklWWW1xd0xoaXJtbUhtQVY2TmlSdDBidTIzekdjcX
Q4OXhRdDVUT3FKMVdMc3grZW4xQlBIVEd5NmFRWQp3MDNheGYrd0dvOVVPMS9DdjdBNVBqUEV3dkNNVlZqRWIraXd2VWhPVnVoeTEwV3RtM0Fq
WVMrZ1JSbjJrSmRwCnFVKy84QjZVdXZlcVVGRmQ4cnU3MW5jWTVlV1lwUGFGWUNkY3k5NHAyU2V3VTBuSUsxd0VBelQyUE1tZUh4T1EKSFB3Nm
dDdHZyOTd2T25SUllHakdSRFZEOHNUTHJsTnNzNENFQzVhNm56RHhlcXVMc0lucCtlMU5JM1VOU1lZTwpoa3UwMTNKbWlMNWtXU1hGMWVST0Vn
PT08L1NpZ25hdHVyZVZhbHVlPgo8S2V5SW5mbz4KPEtleVZhbHVlPgo8UlNBS2V5VmFsdWU%2BCjxNb2R1bHVzPgp6VG9mSHBXQWRoSDNCUi8r
MWxWVk5HUlZZMnFIM0g0KzhjRGFvZmc1Z3k2b2F6Z0IvcVZUWml4bStldVpGMXdWCmEvVDVTUjBDQmVGRjRKWUJtQzBIV2wzOWIyYnFvTkdWME
lMTEt5akRyRTg4cEhQK2s1UEJGZWI5OHpSQVk5NWYKUERPUGZnRmM0ZzY0Vzc2ZnZyaThxZlh4MzY2NVVBVE9UWG52cW5GT25pbEEvTWw5MDB1
c3Q1RHkvSUt5R2dWVAo0eGdtMm5WUUQ2SFltZzdSanlnYS9MQnRURWVLZ2MzaysrZk01dDhBemhkb05DaUdaL0V6MVJ6dGFuakVvQnpXCmRTcm
1IQUdzZW1NVXhGTFBwUUo4eWdsSVlpTDdmRWt5UTBLTXZSY1REazBwVnptTkVxVE5LUTNtUHdwTXorVFcKTTgrd01jOUZqTnRaYUdjMjEzb21X
UT09CjwvTW9kdWx1cz4KPEV4cG9uZW50PgpBUUFCCjwvRXhwb25lbnQ%2BCjwvUlNBS2V5VmFsdWU%2BCjwvS2V5VmFsdWU%2BCjwvS2V5SW5m
bz4KPC9TaWduYXR1cmU%2BPHNhbWxwOlN0YXR1cz48c2FtbHA6U3RhdHVzQ29kZSBWYWx1ZT0idXJuOm9hc2lzOm5hbWVzOnRjOlNBTUw6Mi4w
OnN0YXR1czpTdWNjZXNzIi8%2BPC9zYW1scDpTdGF0dXM%2BPHNhbWw6QXNzZXJ0aW9uIFZlcnNpb249IjIuMCIgSUQ9Il83NTIyRjdBNUU3OD
JFQTBDOEM1RUVBNUQxN0Y0NzlENyIgSXNzdWVJbnN0YW50PSIyMDE4LTA3LTExVDE0OjQ1OjE3WiI%2BPHNhbWw6SXNzdWVyPmh0dHA6Ly9pZH
A1L21ldGFkYXRhPC9zYW1sOklzc3Vlcj48U2lnbmF0dXJlIHhtbG5zPSJodHRwOi8vd3d3LnczLm9yZy8yMDAwLzA5L3htbGRzaWcjIj4KPFNp
Z25lZEluZm8%2BCjxDYW5vbmljYWxpemF0aW9uTWV0aG9kIEFsZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMS8xMC94bWwtZXhjLWMx
NG4jIi8%2BCjxTaWduYXR1cmVNZXRob2QgQWxnb3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAwLzA5L3htbGRzaWcjcnNhLXNoYTEiLz4K
PFJlZmVyZW5jZSBVUkk9IiNfNzUyMkY3QTVFNzgyRUEwQzhDNUVFQTVEMTdGNDc5RDciPgo8VHJhbnNmb3Jtcz4KPFRyYW5sZm9ybSBBbGdvcm
l0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvMDkveG1sZHNpZyNlbnZlbG9wZWQtc2lnbmF0dXJlIi8%2BCjxUcmFuc2Zvcm0gQWxnb3JpdG
htPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzEwL3htbC1leGMtYzE0biMiLz4KPC9UcmFuc2Zvcm1zPgo8RGlnZXN0TWV0aG9kIEFsZ29yaXRo
bT0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnI3NoYTEiLz4KPERpZ2VzdFZhbHVlPlpMdnFpbmQ1SnRpOEtjbG1OZXptdEhZMV
JaMD08L0RpZ2VzdFZhbHVlPgo8L1JlZmVyZW5jZT4KPC9TaWduZWRJbmZvPgo8U2lnbmF0dXJlVmFsdWU%2BTkY5eEp3QWFod0hzTllmRjdLQl
hCWmpSZDhPS1U0a2xSMDBadzlWbExnQ1MyMktRUGtFT0l1K3lZNlBFaDRGVQpOQU11eE5jcnhSSFRjWkxGVnY0a2llalVBMVo3clQ0dVU1WnlO
Umptem1vaVA5Z1NDU2MrWVJYYmw4b3VVbi8xCjRKaVY4d25ZWVVsa2pDTW5wYlFwa3p6V0M0eFlXUkc5cUQzT0REOTFKMEJ5WXZKYXZPRkF3QW
JDS0E1bU1YOE8KZHdEaUVhdzhJdjVGTlMvbjdiejloSlJkdFhkbCtod1JIMWE0dUxLdEgvd2tZRXY1K2Z3MjZxRC8vU2orakxacgpTQmVGeG03
WXp5ZmxhVmFOL3NjWmNYemZLQkt2TGFJVG5VNzJzUFd6R1l1TlJ0QWZoeDJ0OHhnMTRiTkxQTU1XCmtOZHNCa3ZacU02d3E3aUI4eml2aVE9PT
wvU2lnbmF0dXJlVmFsdWU%2BCjxLZXlJbmZvPgo8S2V5VmFsdWU%2BCjxSU0FLZXlWYWx1ZT4KPE1vZHVsdXM%2BCnpUb2ZIcFdBZGhIM0JSLy
sxbFZWTkdSVlkycUgzSDQrOGNEYW9mZzVneTZvYXpnQi9xVlRaaXhtK2V1WkYxd1YKYS9UNVNSMENCZUZGNEpZQm1DMEhXbDM5YjJicW9OR1Yw
SUxMS3lqRHJFODhwSFArazVQQkZlYjk4elJBWTk1ZgpQRE9QZmdGYzRnNjRXNzZmdnJpOHFmWHgzNjY1VUFUT1RYbnZxbkZPbmlsQS9NbDkwMH
VzdDVEeS9JS3lHZ1ZUCjR4Z20yblZRRDZIWW1nN1JqeWdhL0xCdFRFZUtnYzNrKytmTTV0OEF6aGRvTkNpR1ovRXoxUnp0YW5qRW9CelcKZFNy
bUhBR3NlbU1VeEZMUHBRSjh5Z2xJWWlMN2ZFa3lRMEtNdlJjVERrMHBWem1ORXFUTktRM21Qd3BNeitUVwpNOCt3TWM5RmpOdFphR2MyMTNvbV
dRPT0KPC9Nb2R1bHVzPgo8RXhwb25lbnQ%2BCkFRQUIKPC9FeHBvbmVudD4KPC9SU0FLZXlWYWx1ZT4KPC9LZXlWYWx1ZT4KPC9LZXlJbmZvPg
o8L1NpZ25hdHVyZT48c2FtbDpTdWJqZWN0PjxzYW1sOk5hbWVJRCBGb3JtYXQ9InVybjpvYXNpczpuYW1lczp0YzpTQU1MOjIuMDpuYW1laWQt
Zm9ybWF0OnBlcnNpc3RlbnQiIE5hbWVRdWFsaWZpZXI9Imh0dHA6Ly9pZHA1L21ldGFkYXRhIiBTUE5hbWVRdWFsaWZpZXI9Imh0dHA6Ly90ZX
N0c2VydmVyL21ldGFkYXRhLyI%2BXzA2MUM3OEQ4NkU2RTkxNTREMTUyNzA2OERBQkZGQjM3PC9zYW1sOk5hbWVJRD48c2FtbDpTdWJqZWN0Q2
9uZmlybWF0aW9uIE1ldGhvZD0idXJuOm9hc2lzOm5hbWVzOnRjOlNBTUw6Mi4wOmNtOmJlYXJlciI%2BPHNhbWw6U3ViamVjdENvbmZpcm1hdG
lvbkRhdGEgTm90T25PckFmdGVyPSJGSVhNRSIgUmVjaXBpZW50PSJodHRwOi8vdGVzdHNlcnZlci9sb2dpbi8iIEluUmVzcG9uc2VUbz0iXzdB
NzY1OEI0MkIwRjc5QUM5OTNGMjlBMDRBRjA5NUZEIi8%2BPC9zYW1sOlN1YmplY3RDb25maXJtYXRpb24%2BPC9zYW1sOlN1YmplY3Q%2BPHNh
bWw6Q29uZGl0aW9ucyBOb3RCZWZvcmU9IkZJWE1FIiBOb3RPbk9yQWZ0ZXI9IkZJWE1FIj48c2FtbDpBdWRpZW5jZVJlc3RyaWN0aW9uPjxzYW
1sOkF1ZGllbmNlPmh0dHA6Ly90ZXN0c2VydmVyL21ldGFkYXRhLzwvc2FtbDpBdWRpZW5jZT48L3NhbWw6QXVkaWVuY2VSZXN0cmljdGlvbj48
L3NhbWw6Q29uZGl0aW9ucz48c2FtbDpBdXRobN0YXRlbWVudCBBdXRobkluc3RhbnQ9IkZJWE1FIiBTZXNzaW9uSW5kZXg9Il183NTIyRjdBNU
U3ODJFQTBDOEM1RUVBNUQxN0Y0NzlENyI%2BPHNhbWw6QXV0aG5Db250ZXh0PjxzYW1sOkF1dGhuQ29udGV4dENsYXNzUmVmPnVybjpvYXNpcz
puYW1lczp0YzpTQU1MOjEuMDphbTpwYXNzd29yZDwvc2FtbDpBdXRobkNvbnRleHRDbGFzc1JlZj48L3NhbWw6QXV0aG5Db250ZXh0Pjwvc2Ft
bDpBdXRoblN0YXRlbWVudD48L3NhbWw6QXNzZXJ0aW9uPjwvc2FtbDpSZXNwb25zZT4B&RelayState=446b1a4e-c9d5-4de2-b247-0d55
2177ab2b'
(Pdb) provider_id = lasso.profileGetIssuer(payload.decode('utf-8'))
(Pdb) provider_id
(Pdb)
```

Après un coup de débogage du message SAML, celui-ci est bien valide (et contient donc bien un <saml:Issuer>).

Et après une relecture rapide du code de liblasso, je ne comprends pas comment, à partir d'un tel contenu, lasso_profile_get_issuer peut renvoyer

NULL.

Je continue de chercher.

#147 - 11 juillet 2018 17:12 - Frédéric Péters

Avant de continuer à perdre du temps sur une mauvaise version de lasso, tu peux en donner le commit ?

#148 - 11 juillet 2018 17:16 - Paul Marillonnet

Paul Marillonnet a écrit :

> Je suis sur liblasso3 2.6.0.7.gf33d-1~eob90+1.

J'imagine, par concordance des 4 premiers caractères du hachage, que c'est f33d51db53373eb1f0a6429320e9de60210d5270, non ?

#149 - 11 juillet 2018 17:40 - Frédéric Péters

Alors tu refais la même danse, création d'un fichier de test minimal, ticket côté lasso, etc.

#150 - 11 juillet 2018 18:00 - Paul Marillonnet

Hop #25220.

#151 - 12 juillet 2018 14:53 - Benjamin Dauvergne

Ok c'est entièrement ma faute, j'ai indiqué n'importe quoi à Paul, ce n'est pas request.META.get('QUERY_STRING') or request.body qu'il faut faire mais:

```
if request.method == 'GET':
    payload = request.META.get('QUERY_STRING') or ''
elif request.method == 'POST':
    payload = request.POST.get('SAMLResponse') or ''
# ou éventuellement si on sait qu'on est sur un endpoint SOAP
    payload = request.body
```

Désolé pour ma connerie (on peut confondre l'encodage du binding HTTP Post avec HTTP Redirect, la différence c'est la compression du champ SAMLResponse/Request).

#152 - 12 juillet 2018 15:05 - Paul Marillonnet

Pas de souci, merci pour l'info, je reprends ça.

#153 - 12 juillet 2018 16:39 - Paul Marillonnet

Benjamin Dauvergne a écrit :

> Un truc que je disais plus haut pour ne pas faire de réessai mais que tu as du rater:
>
> > Je note aussi une nouvelle API dans lasso dont tu vas pouvoir te servir:
> >
> > [...]
> >
> > qui te donnera directement l'entityID de la requête entrante sans devoir faire un processRequestMsg.

J'ai viré les process*Msg et récupéré les entityIDs avec la nouvelle API, mais je me mange des ProfileMissingResponseError. Je vais regarder le code des process*Msg pour voir si ces méthodes font autre chose que de récupérer dans login l'entityID de la requête.

#154 - 12 juillet 2018 16:44 - Paul Marillonnet

Par exemple, dans lasso.git/lasso/id-ff/login.c:2241, la fonction lasso_login_process_request_msg fait pas mal d'autres choses que récupérer l'entity ID et initialiser le login.remoteProviderId.

Ça me paraît un peu saugrenu de devoir prendre en charge ça dans les vues mellon :)

#155 - 12 juillet 2018 16:51 - Frédéric Péters

Le code de traitement des parcours SSO il est déjà en place et fonctionne, tu ne dois pas t'y perdre. Ce ticket il me semble appeler juste à faire un getIssuer au début, trouver les métadonnées qui correspondent (depuis un fichier de fédérations), les ajouter via addProvider et laisser le code actuel reprendre là.

#156 - 18 juillet 2018 13:20 - Paul Marillonnet

- Fichier 0001-support-federation-file-loading-19396.patch ajouté

- Statut changé de Nouveau à Solution proposée

- Patch proposed changé de Non à Oui

Frédéric Péters a écrit :

> Le code de traitement des parcours SSO il est déjà en place et fonctionne, tu ne dois pas t'y perdre. Ce ticket il me semble appeler juste à faire un getIssuer au début, trouver les métadonnées qui correspondent (depuis un fichier de fédérations), les ajouter via addProvider et laisser le code actuel reprendre là.

Merci, oui, je m'y étais perdu.

Le patch à jour avec les parcours SSO inchangés.
J'ai utilisé la nouvelle API pour LoginView.continue_sso_artifact, mais après coup je ne sais pas si elle a sa place ici, elle ne semble pas prendre en charge la résolution d'artefact pour retrouver l'identifiant du fournisseur. Il faudrait peut-être laisser la double capture d'exception proposée précédemment, juste pour cette méthode de la CBV LoginView ?
(En l'état actuel, pour cette vue, le getIssuer ne renvoie rien, et l'ensemble de fournisseurs définis dans la conf sont chargés dans le serveur lasso. Tout ça alors qu'on retrouve, plus tard dans le code de cette fonction, l'identifiant du fournisseur concerné.)

Autre chose encore, je ne sais pas comment découper clairement le patch pour améliorer la lisibilité, peut-être :
1. les ajouts dans utils et le nouveau module federation_utils
2. les modifs dans le reste du code existant de mellon
3. les jeux de données pour les tests
4. les tests

Dernière chose, le code ne tourne qu'au dessus du patch proposé mais non validé dans #24531.

#157 - 18 juillet 2018 13:30 - Paul Marillonnet

- Lié à Bug #24531: crash tests tox avec lasso 2.6.0-1~eob90+1 ajouté

#158 - 18 juillet 2018 15:44 - Benjamin Dauvergne

Paul Marillonnet a écrit :

> Frédéric Péters a écrit :
>
> > Le code de traitement des parcours SSO il est déjà en place et fonctionne, tu ne dois pas t'y perdre. Ce ticket il me semble appeler juste à faire un getIssuer au début, trouver les métadonnées qui correspondent (depuis un fichier de fédérations), les ajouter via addProvider et laisser le code actuel reprendre là.
>
> Merci, oui, je m'y étais perdu.
>
> Le patch à jour avec les parcours SSO inchangés.
> J'ai utilisé la nouvelle API pour LoginView.continue_sso_artifact, mais après coup je ne sais pas si elle a sa place ici, elle ne semble pas prendre en charge la résolution d'artefact pour retrouver l'identifiant du fournisseur. Il faudrait peut-être laisser la double capture d'exception proposée précédemment, juste pour cette méthode de la CBV LoginView ?

Non la résolution d'artefact c'est encore un cas particulier à la con, voir le code dans authentic de résolution d'un fournisseur SAML via un artifact (http://git.entrouvert.org/authentic.git/tree/src/authentic2/saml/managers.py#n57):

- dans LibertyProvider on conserve le hash SHA1 de l'entityID au format hexadécimal (attribut entity_id_sha1)
- un artifact est une chaîne encodé base64 de la série d'octets suivant (voir code dans lasso http://git.entrouvert.org/lasso.git/tree/lasso/saml-2.0/profile.c#n195)
  0. 00
  1. 04
  2 à 3. l'index du endpoint de résolution d'artifact encodé sur 16bits big-endian (OSEF ici, c'est Lasso qui fera la recherche plus loin, quand on lui aura donné les métadonnées)
  4 à 23. le hash SHA1 de l'entityID
  24 à 43. une séquence aléatoire

Le plus simple c'est de s'inspirer du code du manager dans A2, tu devras conserver quelque part le hash sha1 des entityID (avec des liens symboliques vers tes métadonnées par exemple), puis tu décodes l'artifact, tu extrais les octets 4 à 23 (decoded[4:24]) et tu recherches le bon entity_id à partir de cela.

> (En l'état actuel, pour cette vue, le getIssuer ne renvoie rien, et l'ensemble de fournisseurs définis dans la conf sont chargés dans le serveur lasso. Tout ça alors qu'on retrouve, plus tard dans le code de cette fonction, l'identifiant du fournisseur concerné.)

Oui ce n'est pas possible de générer l'entityID à partir d'un artifact sans stocker un mapping sha1->entityID quelque part.

> Autre chose encore, je ne sais pas comment découper clairement le patch pour améliorer la lisibilité, peut-être :
> 1. les ajouts dans utils et le nouveau module federation_utils
> 2. les modifs dans le reste du code existant de mellon
> 3. les jeux de données pour les tests
> 4. les tests
>
> Dernière chose, le code ne tourne qu'au dessus du patch proposé mais non validé dans #24531.

J'ai validé ton patch.

#159 - 19 juillet 2018 12:22 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

Merci, j'ai commencé à regarder ça.
Ne poussant pas encore mes commits dans une branche à part, je mets le patch WIP ici que je vais retoucher plus tard aujourd'hui.

La solution des liens symboliques ne m'inspirait pas trop à première vue, je voudrais opter pour un fichier de dump json d'un dictionnaire {<empreinte sha1>: <entity ID>}
(cf les fonctions get_entity_id_from_fingerprint et fingerprint_mapping_single_update du module federation_utils)

#160 - 19 juillet 2018 12:23 - Paul Marillonnet

- Statut changé de Solution proposée à En cours

- Patch proposed changé de Oui à Non

#161 - 19 juillet 2018 13:04 - Benjamin Dauvergne

Paul Marillonnet a écrit :

> Merci, j'ai commencé à regarder ça.
> Ne poussant pas encore mes commits dans une branche à part, je mets le patch WIP ici que je vais retoucher plus tard aujourd'hui.
>
> La solution des liens symboliques ne m'inspirait pas trop à première vue, je voudrais opter pour un fichier de dump json d'un dictionnaire {<empreinte sha1>: <entity ID>}
> (cf les fonctions get_entity_id_from_fingerprint et fingerprint_mapping_single_update du module federation_utils)

La désérialisation JSON c'est un truc lent, beaucoup plus lent qu'un simple open().

Tu veux remplacer metadata = open(os.path.join(metadata_dir, entityid_sha1)).read() par

```
d = json.load(open(os.path.join(metadata_dir, 'entityid_sha1_map.json'))
entityid = d[entityid_sha1]
metadata = open(os.path.join(metadata_dir, slugify(entity_id1) + '.xml')).read()
```

#162 - 19 juillet 2018 17:58 - Paul Marillonnet

Je suis parti sur une solution où le mapping se fait à l'aide d'un ensemble de fichiers dans un sous-répertoire du répertoire de cache des métadonnées.
Pour chaque fichier, le nom est l'empreinte sha1 en base 32, et le contenu est l'entity ID.

Concernant les vues je suis dans le doute, encore une fois à cause de mon manque de connaissance de SAML.
Par exemple test_create_server_good_metadata_url :

```
def test_create_server_good_metadata_url(mocker, rf,
 private_settings, caplog):
        private_settings.MELLON_IDENTITY_PROVIDERS = [
            {
                'METADATA_URL': 'http://example.com/metadata',
            }
        ]

        request = rf.get('/')
...
```

Rien dans le rf.get('/') ne donne d'info sur l'issuer, donc, lors de la réception de la requête par mellon, le profileGetIssuer ne peut pas retrouver le provider_id à partir de la requête, non ?
Et pourtant, un fournisseur doit bien être chargé par mellon pour effectuer le SSO, non ?

#163 - 19 juillet 2018 18:23 - Benjamin Dauvergne

Je ne sais pas de quoi tu parles, rf c'est une RequestFactory ça crée des objets request, ça n'exécute pas le code de l'application. Ensuite ce test

vérifie juste qu'on arrive à créer un objet LassoServer dans l'ancien modèle purement statique. Si ce modèle n'a pas complètement disparu avec tes changements c'est du code à peu près toujours valable.

#164 - 19 juillet 2018 18:29 - Paul Marillonnet

Oui, je vais dérouler le code sans les changements, pour comprendre ce que j'ai cassé.

#165 - 25 juillet 2018 10:29 - Paul Marillonnet

- Fichier 0001-support-federation-file-loading-19396.patch ajouté

- Statut changé de En cours à Solution proposée

- Patch proposed changé de Non à Oui

Paul Marillonnet a écrit :

> Oui, je vais dérouler le code sans les changements, pour comprendre ce que j'ai cassé.

Voilà.

Edit: je vais encore tester en long et en large sur mon installation locale.

#166 - 25 juillet 2018 10:52 - Paul Marillonnet

- Statut changé de Solution proposée à En cours

- Patch proposed changé de Oui à Non

#167 - 25 juillet 2018 11:37 - Paul Marillonnet

- Fichier 0001-support-federation-file-loading-19396.patch ajouté

- Statut changé de En cours à Solution proposée

- Patch proposed changé de Non à Oui

Paul Marillonnet a écrit :

> Edit: je vais encore tester en long et en large sur mon installation locale.

Testé, et corrigé un oubli de l'intégration de la nouvelle API dans mellon.views.LogoutView::sp_logout_request.

#168 - 26 juillet 2018 14:04 - Benjamin Dauvergne

- plutôt que des try: f = open()..finally: f.close() utiliser with open(...) as f: qui fait pareil et est un idiome Python
- le log ne correspond pas à l'erreur (et url est non défini):

```
def store_fingerprint(entity_id):
    """Adds an entry in the <sha1(entity_id), entity_id> mapping.
    The mapping cache file may be created in not already existing.
    """
    logger = logging.getLogger(__name__)
    m = sha1()
    m.update(entity_id.encode('utf-8'))
    fingerprint = m.digest()
    fingerprint_b32 = base64.b32encode(fingerprint)
    unix_path = default_storage.path(os.path.join('metadata-cache/fingerprints/', fingerprint_b32.decode('
utf-8')))
    dirname = os.path.dirname(unix_path)
    try:
        if os.path.exists(unix_path):
            logger.info('Could\'nt fetch %r', url)
            return
```

  en fait je ne sais pas trop à quoi sert ce test d'existence en plus, si on stocke l'empreinte, alors on la stocke ,si il y en a déjà une tant pis on écrase. Pas trop sûr que les cas d'erreur soient vraiment cernés ici, en général en cas d'erreur pas évidente il faut au moins un commentaire, si tu as du mal à trouver le commentaire à mettre c'est que certainement la raison d'émettre une erreur n'est pas clair.

- vu que l'usage de get_idps() devient super lourd avec les fédérations il faudrait vraiment en limiter l'usager (parser 5Mo de XML pour en sortir une liste d'entityID c'est lourd), je vois plusieurs usages restant
  - dans PassiveAuthenticationMiddleware, peu de chance qu'on utilise ça avec des fédérations de toute façon, c'est pensé dans le cadre de

Publik où il n'y a qu'un IDP,
- dans create_loaded_server() si on a pas trouvé de remote_provider_id, à mon avis il ne faut pas faire ça, si on ne trouve pas de remote_providerid, on remonte une erreur et on arrête. Le cas du binding artifact doit être pris en compte ici je pense, plutôt que la double chargement que je vois avec ou sans artifact. Pour prendre en compte le fait que le cache des métadonnées n'est pas encore chargé à la rigueur je regarderai le timestamp posé sur le répertoire metadata-cache et si ça a plus d'1h et que l'entityID est inconnu je tente un parsing/rechargement du fichier complet; idem si le répertoire est vide ou inexistant. Aussi tu ne gères pas les SAMLRequest que les réponse, je ne vois pas trop comment ça prend en compte les requêtes de logout. Ça donnerait un truc du genre

```
def create_server(request):
    entity_id = None
    payload = None
    if request.method  == "GET":
       if 'SAMLResponse' in request.GET or 'SAMLRequest' in request.GET:
           payload = request.META['QUERY_STRING']
       elif 'SAMLart' in request.GET:
           entity_id = get_entity_id_from_artifact(request.GET['SAMLart']) <-- ici si on a pas initialisé meta
data-cache on le fait ou si ça date
    elif 'SAMLResponse' in POST:
       payload = request.POST['SAMLResponse']
    elif 'SAMLRequest' in POST:
       payload = request.POST['SAMLRequest']
    if not entity_id and payload:
       entity_id = lasso_profile_get_issuer(payload)
    ....
    utild.add_service_provider(server, entity_id) <-- idem ici si metadata-cache pas initialisé ou trop view o
n recharge tout via get_idps() mais après on charge que le service visé pas tout le monde
```

#169 - 26 juillet 2018 17:11 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

Merci pour la relecture.
Commencé à intégrer les changements.
Je pose le patch WIP ici, et je continue plus tard.

#170 - 01 août 2018 10:36 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

- Statut changé de Solution proposée à En cours

- Patch proposed changé de Oui à Non

Implémenté tes demandes de correction, il me manque un test à faire passer, et d'autres à écrire aussi je crois. Je mets le patch WIP ici pour mon propre suivi perso, je reviens dessus cet ApM.

#171 - 01 août 2018 18:13 - Paul Marillonnet

- Fichier 0001-WIP-support-federation-file-loading-19396.patch ajouté

On s'approche :)

#172 - 02 août 2018 09:59 - Paul Marillonnet

- Fichier 0001-support-federation-file-loading-19396.patch ajouté

- Statut changé de En cours à Solution proposée

- Patch proposed changé de Non à Oui

#173 - 13 février 2019 19:50 - Benjamin Dauvergne

- Priorité changé de Normal à Bas

- Version cible 1.2.35 supprimé

N'est plus à l'ordre du jour vu qu'on a perdu l'AO Condorcet.

#174 - 13 février 2019 19:50 - Benjamin Dauvergne

- Statut changé de Solution proposée à Nouveau

- Assigné à Paul Marillonnet supprimé

Fichiers

0001-WIP-support-federation-file-loading-19396.patch	5,24 ko	16 octobre 2017	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	28,8 ko	19 octobre 2017	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	29,9 ko	14 novembre 2017	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	30,5 ko	14 novembre 2017	Paul Marillonnet
0001-WIP-add-federation_utils-module-19396.patch	3,64 ko	17 novembre 2017	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	35,5 ko	22 novembre 2017	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	113 ko	23 novembre 2017	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	57,2 ko	24 novembre 2017	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	58,3 ko	24 novembre 2017	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	58 ko	25 novembre 2017	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	57,3 ko	25 novembre 2017	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	57 ko	25 novembre 2017	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	65 ko	28 novembre 2017	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	70,3 ko	28 novembre 2017	Paul Marillonnet
auth_samlresponse.xml	3,95 ko	30 novembre 2017	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	78,5 ko	30 novembre 2017	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	78,6 ko	04 décembre 2017	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	79 ko	05 décembre 2017	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	81,2 ko	07 décembre 2017	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	84,1 ko	05 janvier 2018	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	85,1 ko	05 janvier 2018	Paul Marillonnet
inbetween.diff	9,08 ko	11 janvier 2018	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	86 ko	11 janvier 2018	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	86,2 ko	11 février 2018	Benjamin Dauvergne
0001-WIP-support-federation-file-loading-19396.patch	84,3 ko	14 février 2018	Paul Marillonnet
0002-WIP-federation-file-lazy-provider-loading-19396.patch	14,1 ko	18 avril 2018	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	84,3 ko	18 avril 2018	Paul Marillonnet
0001-federation-file-loading-19396.patch	102 ko	31 mai 2018	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	100 ko	27 juin 2018	Paul Marillonnet
0001-support-federation-file-loading-19396.patch	94 ko	18 juillet 2018	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	100 ko	19 juillet 2018	Paul Marillonnet
0001-support-federation-file-loading-19396.patch	99,4 ko	25 juillet 2018	Paul Marillonnet
0001-support-federation-file-loading-19396.patch	100 ko	25 juillet 2018	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	101 ko	26 juillet 2018	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	101 ko	01 août 2018	Paul Marillonnet
0001-WIP-support-federation-file-loading-19396.patch	101 ko	01 août 2018	Paul Marillonnet
0001-support-federation-file-loading-19396.patch	101 ko	02 août 2018	Paul Marillonnet