# Passerelle - Development #39431

## ajout d'un connecteur "chiffrement de fichier"

31 janvier 2020 00:34 - Thomas Noël

| | | | |
|---|---|---|---|
| **Statut:** | Fermé | **Début:** | 31 janvier 2020 |
| **Priorité:** | Normal | **Echéance:** | |
| **Assigné à:** | Thomas Noël | **% réalisé:** | 0% |
| **Catégorie:** | | **Temps estimé:** | 0:00 heure |
| **Version cible:** | | | |
| **Patch proposed:** | Oui | **Planning:** | Non |

**Description**

Éléments de configuration de l'instance :

- une clé publique et/ou une clé privée, format PEM
- decrypt_url_base: URL de base du système de déchiffrement, sera l'URL du endpoint "decrypt" sur la passerelle de déchiffrement

Endpoints:

- encrypt
  - reçoit dans un dictionnaire un fichier au "format wcs" : {"file": {filename:..., content_type:..., content:...base64...}}
  - chiffre content avec la clé publique, stocke le résultat sur le disque, lui attribue un uuid.uuid4()
  - retient aussi filename et content_type ainsi que la date d'arrivée
  - renvoie au moins un "redirect_url" construit selon le modèle « decrypt_url_base/<uid> »

- decrypt
  - reçoit l'uid du fichier à déchiffrer (en pattern)
  - renvoie le payload déchiffré, avec content-type et filename (ie ne renvoie pas de json mais le fichier lui-même)
  - ne répond que sur la clé privée est configurée, sinon renvoyer un plain/text genre "no encrypt key"
  - log de l'accès dans un table log spécifique (uuid, IP, date, api_user)

Les endpoints ne sont bien sûr pas ouverts (perm='can_access')

NB : idéalement, une fois stockée, ça serait bien que la clé privée ne soit pas affichée dans l'interface d'édition de l'instance. On afficherait uniquement son empreinte, ainsi que cette de la clé publique, pour vérification.

### Révisions associées

**Révision e137d66d - 04 mars 2020 12:00 - Thomas Noël**

add cryptor connector (#39431)

### Historique

**#1 - 31 janvier 2020 00:35 - Thomas Noël**

*- Assigné à mis à Thomas Noël*

Pour le chiffrement/déchiffrement : https://www.pycryptodome.org/en/latest/src/examples.html#encrypt-data-with-rsa

**#2 - 24 février 2020 23:23 - Thomas Noël**

*- Fichier 0001-add-cryptor-connector-39431.patch ajouté*

*- Statut changé de Nouveau à Solution proposée*

*- Patch proposed changé de Non à Oui*

Voici donc la proposition qui fait le taf. J'ai passé une bonne petite à rendre l'affaire compatible python-2-et-3 parce que c'est pas mon fort.

Pour chaque fichier chiffré, on a son contenu dans un fichier <uuid> et ses métadonnées (nom, content-type, date de création) dans un <uuid>.json, dans /media/cryptor/<slug>/ab/cd/ (où ab et cd sont les 4 premiers caractères de l'uuid)

Note : le modèle CryptedFile ne sert pas vraiment, en dehors de générer un uuid "certainement unique". Ca pourrait cependant servir plus tard, alors je l'ai laissé.

Par rapport à la "spec", pas de suivi des accès, ils sont déjà dans les logs nginx.

(Fonctionne comme attendu via un RemoteOpaqueUploadStorage de wcs)

#### #3 - 02 mars 2020 11:03 - Benjamin Dauvergne

Je valide le chiffrement RSA/OAEP/AES/EAX recopié des example de pycryptodome (en général les exemples donnent les bonnes pratiques, donc bonne idée).

Tu peux utiliser passerelle.utils.files.atomic_write plutôt que ça :

```
        with self.named_tempfile() as tpf:
            write_encrypt(tpf, data, self.public_key)
            tpf.flush()
            os.fsync(tpf.file.fileno())
            tempfile_name = tpf.name
        os.rename(tempfile_name, content_filename)
```

voir passerelle/utils/zip.py par exemple:

```
from·passerelle.utils.files·import·atomic_write

...
        with atomic_write(full_path, dir=tmp_dir) as fd:
            self.render_to_file(fd)
```

#### #4 - 02 mars 2020 14:04 - Paul Marillonnet

Petite question ouverte, dont je ne prétends pas avoir la réponse : est-ce qu'on aurait intérêt à l'initialisation de la ressource de faire une rapide vérification que Cipher_{priv}(Cipher_{pub}('abc')) == 'abc', ou vice-versa, pour vérifier que les deux clés publique et privée se complémentent bien ?

Je souhaiterais éviter le cas où, en prod, le connecteur a mal été initialisé, et on ne s'en rend compte que lorsque des demandes ont déjà été soumises, et que des fichiers chiffrés des usagers ne sont plus déchiffrables car aïe dommage la clé privée ne correspond pas.

#### #5 - 02 mars 2020 15:01 - Benjamin Dauvergne

Paul Marillonnet a écrit :

> Petite question ouverte, dont je ne prétends pas avoir la réponse : est-ce qu'on aurait intérêt à l'initialisation de la ressource de faire une rapide vérification que Cipher_{priv}(Cipher_{pub}('abc')) == 'abc', ou vice-versa, pour vérifier que les deux clés publique et privée se complémentent bien ?

À l'instanciation du service on aura généralement pas la clé privée, elle sera conservée par la collectivité sur un service en DMZ auquel on ne devrait pas avoir accès si j'ai bien compris; de leur coté ça peut être intéressant oui.

> Je souhaiterais éviter le cas où, en prod, le connecteur a mal été initialisé, et on ne s'en rend compte que lorsque des demandes ont déjà été soumises, et que des fichiers chiffrés des usagers ne sont plus déchiffrables car aïe dommage la clé privée ne correspond pas.

La solution magique contre la connerie on ne l'a pas encore :)

#### #6 - 03 mars 2020 00:49 - Thomas Noël

Paul Marillonnet a écrit :

> Petite question ouverte, dont je ne prétends pas avoir la réponse : est-ce qu'on aurait intérêt à l'initialisation de la ressource de faire une rapide vérification que Cipher_{priv}(Cipher_{pub}('abc')) == 'abc', ou vice-versa, pour vérifier que les deux clés publique et privée se complémentent bien ?
> Je souhaiterais éviter le cas où, en prod, le connecteur a mal été initialisé, et on ne s'en rend compte que lorsque des demandes ont déjà été soumises, et que des fichiers chiffrés des usagers ne sont plus déchiffrables car aïe dommage la clé privée ne correspond pas.

Yep. Bon, en pratique jamais on mettra les deux clés sur la même instance, hein :) L'idée c'est qu'une passerelle avec la clé privée (decrypt) est installé sur un autre réseau que la passerelle de chiffrement (qui elle fera partie du "publik public")

Mais oui pour l'idée et c'est pour ça que j'écrivais dans la description : « idéalement, une fois stockée, ça serait bien que la clé privée ne soit pas affichée dans l'interface d'édition de l'instance. On afficherait uniquement son empreinte, ainsi que cette de la clé publique, pour vérification »... et puis basta, ça sera vraiment pas compliqué à vérifier au premier test d'un chiffrage/déchiffrage, alors je n'en ai pas fait une priorité.

#### #7 - 03 mars 2020 01:02 - Thomas Noël

*- Fichier 0001-add-cryptor-connector-39431.patch ajouté*

*- Fichier cryptor-interdiff.patch ajouté*

Benjamin Dauvergne a écrit :

> Je valide le chiffrement RSA/OAEP/AES/EAX recopié des example de pycryptodome (en général les exemples donnent les bonnes pratiques, donc bonne idée).
> Tu peux utiliser passerelle.utils.files.atomic_write plutôt que ça

J'avais raté cet outil, merci. Voici le patch adapté (et l'interdiff). C'est également envoyé dans la branche.

#### #8 - 03 mars 2020 09:44 - Paul Marillonnet

Thomas Noël a écrit :

> Yep. Bon, en pratique jamais on mettra les deux clés sur la même instance, hein :) L'idée c'est qu'une passerelle avec la clé privée (decrypt) est installé sur un autre réseau que la passerelle de chiffrement (qui elle fera partie du "publik public")

My bad, après une première lecture du code je pensais que chaque instance disposerait de sa paire de clés publique et privée – et qu'on exploiterait la commutativité de l'exponentiation modulatoire, i.e.

```
Cipher_{privB}(Cipher_{pubA}(Cipher_{pubB}(Cipher_{privA}('abc')))) ==
Cipher_{privB}(Cipher_{pubB}(Cipher_{pubA}(Cipher_{privA}('abc')))) ==
'abc'
```

> Mais oui pour l'idée et c'est pour ça que j'écrivais dans la description : « idéalement, une fois stockée, ça serait bien que la clé privée ne soit pas affichée dans l'interface d'édition de l'instance. On afficherait uniquement son empreinte, ainsi que cette de la clé publique, pour vérification »... et puis basta, ça sera vraiment pas compliqué à vérifier au premier test d'un chiffrage/déchiffrage, alors je n'en ai pas fait une priorité.

Oui ok je comprends (et j'aurais dû lire la description ).

#### #9 - 04 mars 2020 10:53 - Frédéric Péters

Sur le fond, je m'interroge sur redirect_url_base, je trouve "bizarre" que ça se trouve à ce niveau, je me dis que je l'aurais préféré en paramétrage du système de stockage dans wcs; mais n'en faisons rien.

Ajouter à debian/ la dépendance sur python-pycryptodome.

Grosse suggestion aussi :

```
--- a/passerelle/static/css/style.css
+++ b/passerelle/static/css/style.css
@@ -177,6 +177,10 @@ li.connector.dpark a::before {
        content: "\f1b9";  /* car */
 }

+li.connector.cryptor a::before {
+       content: "\f023";  /* lock */
+}
+
 li.connector.status-down span.connector-name::after {
```

contextlib et tempfile importés mais pas/plus utilisés.

```
-                                    verbose_name=_('Encrypt RSA public key (PEM format)'),
+                                    verbose_name=_('Encryption RSA public key (PEM format)'),
...
-                                     verbose_name=_('Decrypt RSA private key (PEM format)'),
+                                     verbose_name=_('Decryption RSA private key (PEM format)'),
...
-        verbose_name = _('Encrypt / Decrypt')
+        verbose_name = _('Encryption / Decryption')

        uuid = str(cfile.uuid)
```

J'ajouterais un commentaire "# get string representation of UUID object" parce que je me suis demandé pourquoi c'était nécessaire et j'ai crains que ça ne soit des bytes et que ça fasse "b'uuid ici'".

#### #10 - 04 mars 2020 11:34 - Thomas Noël

*- Fichier 0001-add-cryptor-connector-39431.patch ajouté*

*- Fichier interdiff.patch ajouté*

Frédéric Péters a écrit :

> Sur le fond, je m'interroge sur redirect_url_base, je trouve "bizarre" que ça se trouve à ce niveau, je me dis que je l'aurais préféré en paramétrage du système de stockage dans wcs; mais n'en faisons rien.

Ca peut tout à fait être revu plus tard, mais ça passerait d'abord par une adaptation dans w.c.s (qommon::upload_storage).

> Ajouter à debian/ la dépendance sur python-pycryptodome.

Yep.

> Grosse suggestion aussi :

Joli, merci pour le \f023 lock

> contextlib et tempfile importés mais pas/plus utilisés.

Retirés.

> J'ajouterais un commentaire "# get string representation of UUID object" parce que je me suis demandé pourquoi c'était nécessaire et j'ai crains que ça ne soit des bytes et que ça fasse "b'uuid ici'".

J'ai un poil pris du temps sur ça, car on reçoit dans cfile.uuid un UUID(truc) et cette classe dispose explicitement de str pour en faire la représentation habituelle (dans le but, pour moi, d'en faire un nom de fichier) :

```
class UUID:

    def __str__(self):
        hex = '%032x' % self.int
        return '%s-%s-%s-%s-%s' % (
            hex[:8], hex[8:12], hex[12:16], hex[16:20], hex[20:])
```

J'ai donc préféré faire cet appel très explicite à str() plutôt que "cacher" cela dans un %s ou autre.

Nouveau patch et interdiff attaché.

#### #11 - 04 mars 2020 11:54 - Frédéric Péters

*- Statut changé de Solution proposée à Solution validée*

> J'ai donc préféré faire cet appel très explicite à str() plutôt que "cacher" cela dans un %s ou autre.

Oui oui, très bien (surtout que dans l'urljoin ça aurait planté), je demandais juste le commentaire parce qu'à la lecture je me suis demandé pourquoi c'était ainsi. (avant de voir que uuid était un objet).

#### #12 - 04 mars 2020 12:00 - Thomas Noël

*- Statut changé de Solution validée à Résolu (à déployer)*

```
commit e137d66def258bb2c30101de68f95d2cf03b7e08
Author: Thomas NOEL <tnoel@entrouvert.com>
Date:   Thu Feb 20 17:19:34 2020 +0100

    add cryptor connector (#39431)
```

#### #13 - 04 mars 2020 15:15 - Frédéric Péters

*- Statut changé de Résolu (à déployer) à Solution déployée*

### Fichiers

| | | | |
|---|---|---|---|
| 0001-add-cryptor-connector-39431.patch | 22,5 ko | 24 février 2020 | Thomas Noël |
| cryptor-interdiff.patch | 2,1 ko | 03 mars 2020 | Thomas Noël |

| | | | |
|---|---|---|---|
| 0001-add-cryptor-connector-39431.patch | 22 ko | 03 mars 2020 | Thomas Noël |
| 0001-add-cryptor-connector-39431.patch | 22,9 ko | 04 mars 2020 | Thomas Noël |
| interdiff.patch | 4,73 ko | 04 mars 2020 | Thomas Noël |