Stage conventionné : 9h37

DELASSUS David

11 novembre 2011

Chef de stage : Nicolas WOCJIK
Durée : 6 mois (le mercredi)
Projet : Solution de partage et de sauvegarde de données dans un réseau LAN

Table des matières

Première partie
Introduction

Dans cette partie, nous décrirons les caractéristiques du projet et les technologies qui seront employées pour le mettre en œuvre.

1 Présentation du projet

Le projet de stage proposé par l'entreprise 9h37 consiste à mettre en place un système de sauvegarde et de sécurisation des données d'un réseau de partage. L'utilisateur devra être apte à sauvegarder ses données simplement sur un service de partage sur le réseau LAN de l'organisme. Ces données devront être sauvegardées et chiffrées sur un espace de stockage invisible au reste du réseau. Enfin, elles devront être envoyées sur un hôte distant pour réaliser des backup réguliers. Au sein de l'entreprise, une interface web permettant de gérer les données chiffrées devra être mise en place.

Ce projet me permettra de développer des compétences en architecture réseau, en administration de système *UNIX/Linux* et en développement web avec *Python/Django*. L'apprentissage de méthode de chiffrement (*GnuPG*) sera nécessaire à la réalisation du projet. Les technologies de partage (*Samba*) et de transfert (*SSH*) seront approfondies.

L'enjeu de ce stage pour l'entreprise sera donc le développement d'une solution simple, efficace et sécurisée pour répondre à la demande de leur client.

La demande à laquelle nous allons répondre peut se décomposer en quatre parties :

1. Partage des données sur un réseau LAN interne à l'organisme.
2. Sauvegarde et chiffrement des données sur un espace de stockage interne à l'organisme.
3. Copie des données chiffrées chez un hôte distant (hébergeur, ...).
4. Interface web de gestion des données chiffrées interne à l'organisme.

1.1 Partage des données

La mise en place du partage de données permettra aux membres de l'organisme de mettre en commun leur travail sur un espace de stockage partagé.

Cela implique la mise en place d'un serveur pour l'organisme.

Le partage doit être simple pour l'utilisateur et accessible à tout le réseau LAN.

1.2 Sauvegarde et chiffrement des données

Les données présentes sur le partage devront être sauvegardées et chiffrées. Il faudra donc maintenir une liste des fichiers chiffrées présent sur l'espace de stockage afin de ne pas chiffrer et sauvegarder plusieurs fois un même fichier.

L'espace de stockage n'est pas accessible sur le réseau LAN.

La sauvegarde et le chiffrement des données devra être périodique pour ne pas surcharger le serveur et donc ne pas perturber le travail en cours dans l'organisme.

1.3 Copie chez un hôte distant

Le transfert devra être sécurisé. Il faudra maintenir une liste des fichiers présents sur l'hôte distant afin de ne pas transférer plusieurs fois un fichier déjà à jour.

De même que pour la sauvegarde et le chiffrement, le transfert devra être périodique afin de ne pas saturer la bande passante de l'organisme.

1.4 Interface web

L'interface web permettra de gérer les données chiffrées (les restaurer en cas de suppression sur le partage, les supprimer, ...).

Celle ci se présentera sous la forme d'un navigateur de fichier classique, avec une arborescence, une vue en icône/liste/... et une gestion simplifié des différentes actions sur les fichiers.

L'accès à cette application web devra être restreint aux membres de l'organisme ayant les autorisations nécessaires pour la manipuler (la méthode d'authentification reste à déterminer).

2 Schémas

2.1 Architecture réseau

Pour l'architecture réseau, nous aurons donc :

LAN Un réseau local à l'organisme.

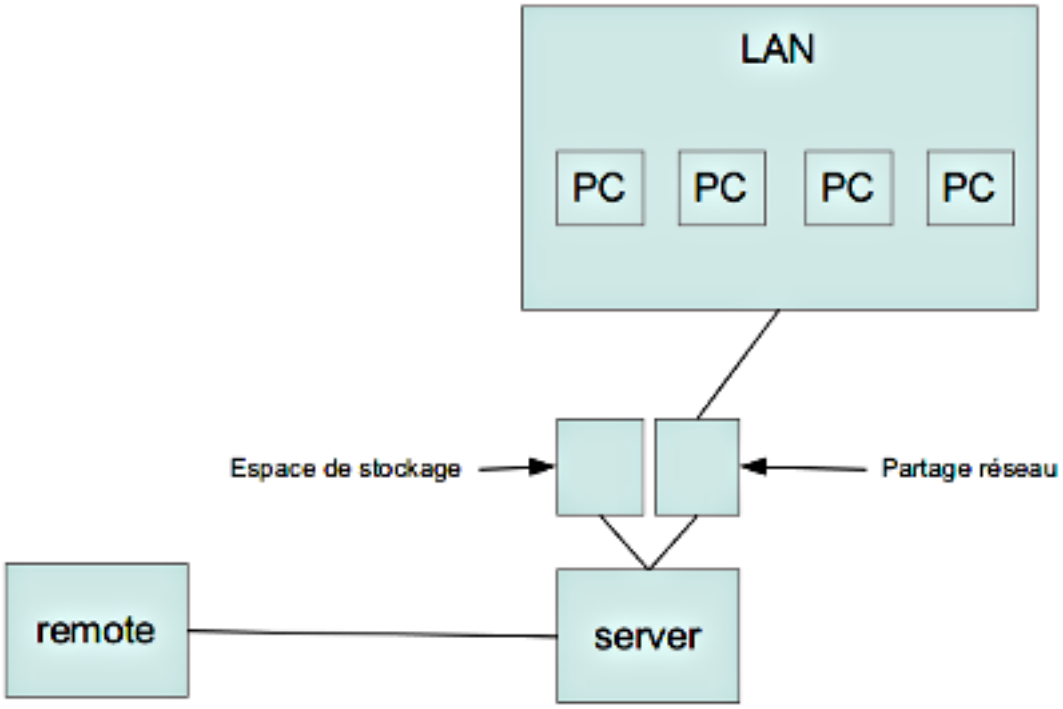
server Un serveur interne à l'organisme.

partage Un partage réseau sur le LAN de l'organisme.

stockage Un espace de stockage interne.

remote Un espace de stockage chez un hôte distant.

Voici un schéma simplifié des besoins du projet :



2.2 Interface web

L'interface web se présentera sous la forme d'un navigateur de fichier classique, elle devra donc comporter les caractéristiques suivantes :

- Arborescence de l'espace de stockage
- Vue en icône (ou liste) de l'espace de stockage
- Informations sur le fichier sélectionné
- Actions groupées (actions sur plusieurs fichiers)

Voici un exemple d'interface possible :

3 Technologies utilisées

Ici, on retrouvera la liste des technologies employées pour réaliser ce projet.

3.1 Serveur

UNIX/Linux : Système d'exploitation du serveur interne de l'organisme.

cron : Utilitaire de programmation des tâches planifiées.

3.2 Partage de données

Samba : Service de partage réseau compatible *UNIX* et *Windows*.

3.3 Sauvegarde et chiffrement

GnuPG : Chiffrement asynchrone des données.

SQLite : Base de données pour maintenir la liste des fichiers présents sur l'espace de stockage.

cp : Utilitaire de copie de données.

3.4 Copie chez un hôte distant

scp : Utilitaire de copie de données via *SSH* (on utilisera des clés RSA, DSA ou EDSA pour l'identification).

SQLite : Base de données pour maintenir la liste des fichiers présents sur l'hôte (on utilisera la base de données présente en locale).

3.5 Interface web

Python : L'interface web sera développé dans ce langage.

Django : On utilisera ce framework pour le développement de l'application web.

Deuxième partie

Compte rendu du stage

4 Développement du script de base pour la sauvegarde et le chiffrement des données

4.1 Convention de l'ANSSI

L'Agence Nationale de la Sécurité des Systèmes d'Information recommande l'utilisation de l'algorithme de hachage **SHA-256**.

4.2 Consigne

Le script de base devra être développé en *Python 2.7* et devra remplir les conditions suivantes :

1. Indentation de 4 espaces, pas de tabulations.
2. Deux répertoires : **src** et **dest**, le fichiers ne doivent pas être copiés de *src* vers *dest*, ce dernier répertoire ne devra contenir que du contenu chiffré.
3. Calculer la somme SHA-256 des fichiers sources et l'afficher.
4. Chiffrer les fichiers avec *GnuPG* dans *dest*.
   (a) Si le fichier n'existe pas, le créer.
   (b) Si le fichier existe le nommer selon ce pattern : *filename.X.gpg* où X varie de 1 à 5 (à la 6è itération, on supprime le fichier).
5. Placer le code source dans un dépôt *git*.
6. Utiliser *unittest* pour les tests unitaires.

4.3 Conclusion

Cette partie du logiciel a été développée sous la forme d'un paquet *Python*.
A l'aide d'un unique objet, on est capable de récupérer la liste des fichiers du répertoire source, de calculer leurs sommes SHA-256 et de les chiffrer via *GnuPG* :

```
from src.filemanager import FileManager

fm = FileManager.FileManager (srcpath, destpath)
filelist = fm.read_entries ()
checksum = fm.hash_entries (filelist)
fm.gpg_encrypt (filelist) # encrypt
```

5 Gestion de la base de données SQL

La base de données SQL sera utilisée pour indexer les fichiers chiffrés sur l'espace de stockage. Grâce à elle, on ne chiffrera pas plusieurs fois un même fichier, et on ne transférera pas sur l'hôte distant plusieurs fois les mêmes données.
Il est donc impératif que la base de données contienne les informations suivantes :

hash : Le hash SHA-256 du fichier non chiffré, il sera utilisé en tant qu'index de la base de données SQL.

path : Le chemin d'accès vers le fichier chiffré, ainsi pour un fichier chiffré on a le hash du fichier non chiffré qui y est associé.

sent : La date à laquelle le fichier a été envoyé sur l'hôte distant (ou 0 s'il n'a pas été envoyé).

Voici par exemple ce que pourra contenir une table :

e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855	try/empty	1320932458
fe19778cf1ce280658154f2b9c01ffbccd825a23460141dcf3794e7a2c0eb629	oops	1315472654
f2ca1bb6c7e907d06dafe4687e579fce76b37e4e93b7605022da52e6ccc26fd2	test	0

On aimerait cependant ne pas avoir à ce soucier du type de la base de données (*MySQL*, *SQLite*, ...), il conviendra donc d'utiliser *Django* pour la gestion de la base de données :

```
from django.db import models

class DatabaseEntry (models.Model):
    checksum = models.CharField (max_length = 64)
    path     = models.CharField (max_length = 256)
    sent     = models.DateTimeField ()
```

6 Intégration Django

L'interface web étant également développé avec *Django*, il convient donc d'intégrer notre script dans une application *Django* qui sera distribuée avec l'interface web.

Le projet, **delikatess**, se présente désormais sous la forme d'un projet *Django* :

webui : L'interface web.

trackfile : Le script de sauvegarde et de chiffrement.

sendit : Le script d'envoie des fichiers chiffrés.

6.1 trackfile

L'application *trackfile* est donc l'intégration de notre script à *Django*. Ainsi la base de données de notre script est commune à celle de *Django* et de notre future application web :

```
>>> from trackfile.utils import FileManager
>>> fm = FileManager (<gpg-key>, <source directory>, <destination directory>, nbackups = 7)
>>> fm.run ()
```

7 Intégration continue

Afin de vérifier la validité du code source produit à chaque modifications, il va falloir développer un script qui va devoir remplir les conditions suivantes :

- Créer un environnement *Python* virtuel (à l'aide de *virtualenv* dans *.venv*).
- Installer les dépendances du projet dans cet environnement (à l'aide du gestionnaire de paquets *Python* : *pip*).
- Exécuter les tests unitaires.