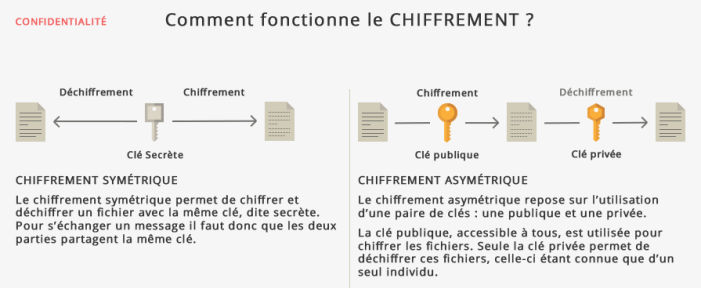
Les Solutions de Chiffrement



Le chiffrement est essentiel pour protéger les données sensibles dans une pile LAMP. Voici une solution complète pour mettre en place le chiffrement dans une pile LAMP, couvrant les aspects suivants : chiffrement des sauvegardes, déchiffrement des fichiers, et chiffrement des communications (HTTPS).

1. Chiffrement des sauvegardes

Pour chiffrer les sauvegardes de bases de données et des fichiers, nous utiliserons GPG (GNU Privacy Guard).

1.1. Installer GPG

```
sudo apt update
sudo apt install gnupg -y
```

1.2. Générer une paire de clés GPG

```
gpg --full-generate-key
```

Suivez les instructions pour configurer le type de clé (en général RSA), la longueur de la clé, et les informations d'identité. Une fois la clé générée, vous obtiendrez deux éléments : - Une **clé privée** (à conserver uniquement

sur le serveur). - Une **clé publique** (à partager avec le client pour qu'il puisse chiffrer les données).

1.3. Exporter la clé publique du serveur

Exportez la clé publique pour qu'elle puisse être partagée avec le client :

```
gpg --export -a "NomUtilisateur" > server-public-key.asc
```

Le fichier `server-public-key.asc` contient la clé publique. Envoyez ce fichier au client en toute sécurité.

1.4. Importer la clé publique sur le client

Sur le côté client, importez la clé publique du serveur pour pouvoir chiffrer les données :

```
gpg --import server-public-key.asc
```

1.5. Chiffrer les données sur le client

Une fois la clé publique importée, le client peut chiffrer des données (comme un fichier) pour qu'elles ne soient lisibles que par le serveur.

Pour chiffrer un fichier `data.txt` :

```
gpg --output data.txt.gpg --encrypt --recipient "NomUtilisateur" data.txt
```

Cela génère un fichier chiffré `data.txt.gpg`, qui peut être transmis en toute sécurité au serveur.

1.6. Modifier les scripts de sauvegarde pour inclure le chiffrement du côté client

Script de sauvegarde de la base de données (`backup_db.sh`) :

```
#!/bin/bash

# Variables
DB_USER="glpiuser"
DB_PASSWORD="password"
DB_NAME="glpidb"
```

```
BACKUP_DIR="/path/to/backup/directory"
DATE=$(date +\%Y-\%m-\%d)
GPG_KEY="your_email@example.com"

# Créer le répertoire de sauvegarde s'il n'existe pas
mkdir -p $BACKUP_DIR

# Effectuer la sauvegarde
mysqldump -u $DB_USER -p$DB_PASSWORD $DB_NAME > $BACKUP_DIR/$DB_NAME-$DATE.sql

# Chiffrer la sauvegarde
gpg --output $BACKUP_DIR/$DB_NAME-$DATE.sql.gpg --encrypt --recipient $GPG_KEY\
 $BACKUP_DIR/$DB_NAME-$DATE.sql

# Supprimer le fichier non chiffré
rm $BACKUP_DIR/$DB_NAME-$DATE.sql

# Supprimer les sauvegardes chiffrées de plus de 7 jours
find $BACKUP_DIR/* -mtime +7 -exec rm {} \;
```

Avant la dernière ligne de commande de suppréssion des sauvegardes chiffrées de plus de 7 jours, on peut ajouter une commande de transfert du fichier chiffré au serveur:

```
scp $BACKUP_DIR/$DB_NAME-$DATE.sql.gpg <UTILISATEUR_SERVEUR>@\
 <ADRESSE_IP_SERVEUR>:<CHEMIN_ESPACE_UTILISATEUR>

# Par exemple: scp $BACKUP_DIR/$DB_NAME-$DATE.sql.gpg osboxes@192.168.1.56:\
# /home/osboxes
```

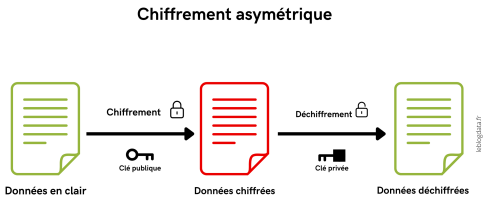
On pourra faire de même pour la solution de script décrite ci-dessous.

Script de sauvegarde des fichiers web (`backup_files.sh`) :

```
#!/bin/bash

# Variables
WEB_DIR="/var/www/html"
BACKUP_DIR="/path/to/backup/directory"
DATE=$(date +\%Y-\%m-\%d)
```

```
GPG_KEY="your_email@example.com"

# Créer le répertoire de sauvegarde s'il n'existe pas
mkdir -p $BACKUP_DIR

# Effectuer la sauvegarde
tar -czf $BACKUP_DIR/html-backup-$DATE.tar.gz $WEB_DIR

# Chiffrer la sauvegarde
gpg --output $BACKUP_DIR/html-backup-$DATE.tar.gz.gpg --encrypt --recipient\
 $GPG_KEY $BACKUP_DIR/html-backup-$DATE.tar.gz

# Supprimer le fichier non chiffré
rm $BACKUP_DIR/html-backup-$DATE.tar.gz

# Supprimer les sauvegardes chiffrées de plus de 7 jours
find $BACKUP_DIR/* -mtime +7 -exec rm {} \;
```

2. Déchiffrement des sauvegardes



1.1. Déchiffrer les données sur le serveur

Une fois le fichier chiffré reçu par le serveur, celui-ci peut le déchiffrer à l'aide de sa clé privée :

```
gpg --output data.txt --decrypt data.txt.gpg
```

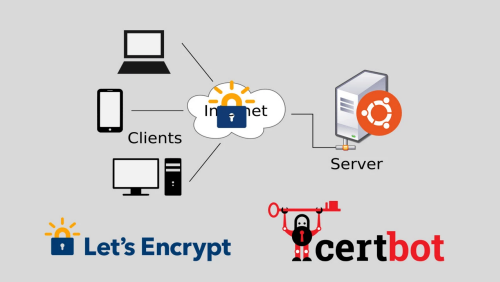
1.2. Résumé de toute l'opération

1. Le **serveur génère** une paire de clés (privée et publique).
2. La **clé publique du serveur est envoyée** au client.
3. Le **client importe** la clé publique du serveur et l'utilise pour **chiffrer les données**.

4. Le **serveur déchiffre** les données avec sa clé privée.

Ainsi, seul le serveur peut lire les données chiffrées envoyées par le client, ce qui garantit une transmission sécurisée.

3. Chiffrement des communications (HTTPS)



Pour chiffrer les communications entre le client et le serveur, vous devez installer un certificat SSL. Vous pouvez utiliser Let's Encrypt, qui fournit des certificats SSL gratuits.

3.1. Installer Certbot

```
sudo apt update
sudo apt install certbot python3-certbot-apache -y
```

3.2. Obtenir un certificat SSL

```
sudo certbot --apache
```

Suivez les instructions pour obtenir et installer le certificat. Certbot configurera automatiquement Apache pour utiliser le certificat SSL.

3.3. Renouvellement automatique

Let's Encrypt les certificats SSL sont valables 90 jours. Certbot crée une tâche cron pour renouveler automatiquement les certificats. Vous pouvez vérifier cette tâche :

```
sudo systemctl list-timers
```

Conclusion

Avec ces étapes, vous avez mis en place une solution de chiffrement complète pour protéger les sauvegardes, les fichiers sensibles et les communications dans votre pile LAMP. Assurez-vous de tester régulièrement vos sauvegardes chiffrées et de renouveler vos certificats SSL pour maintenir la sécurité de votre environnement.