L’administration d’un serveur Apache sous Linux

Module 9 – Sécuriser Apache

Objectifs

- Sécuriser les accès aux fichiers
- Comprendre et mettre en place un certificat SSL
- Connaître les principes de sécurité

Sécuriser Apache

Le Web et la sécurité

- Les sites web sont de plus en plus attaqués
- Les langages dynamiques tels que PHP peuvent créer des failles exploitables par les attaquants
- Le transfert de données sensibles doit être **chiffré** entre le client et le serveur
- Il est important de mettre en place des **bases de sécurité** pour contrer certaines attaques

Sécuriser Apache

Limiter les accès aux fichiers

- Certains fichiers ou répertoires ne doivent pas être accessibles lors de la navigation
- Attention aux **liens symboliques** aussi qui ne doivent pas rediriger vers des répertoires sensibles du système de fichiers
- Les directives de limitation doivent s'appliquer dans les sections **`<Directory>`**, **`<Files>`** et **`<Location>`**.

Sécuriser Apache

Section <Directory>

- Cette section permet de définir des directives sur un répertoire :

```
<Directory /var/www/site1>
   Directive
   Directive
   ...
</Directory>
```

- On peut utiliser des expressions rationnelles :
  `<Directory ~ "^/www/[0-9]{3}>`

Sécuriser Apache

Section <Files>

- Les directives s’appliquent à des fichiers

```
<Files ~ "php$">
   Directive ...
</Files>
```

- S’applique à tous les fichiers se terminant par « php »

Sécuriser Apache

Section <Location>

- Les directives s’appliquent à une URL du site

```
<Location /images>
   Directive ...
</Location>
```

Ordre d’interprétation

- Ces directives peuvent être placées dans la configuration globale, dans la configuration d’un site ou dans un fichier .htaccess

- Apache va interpréter les directives dans un certain ordre :

1. Les sections **`<Directory>`**
2. Les fichiers .htaccess
3. Les sections **`<Files>`**
4. Les sections **`<Location>`**

Sécuriser Apache

Fichier .htaccess

- Un fichier **.htaccess** peut être placé dans tout répertoire et les directives s'appliquent dans le répertoire et ses sous-répertoires
- La multiplication des fichiers .htaccess et leur utilisation peuvent être nuisibles à la sécurité et à la performance des serveurs
- Les directives des fichiers .htaccess peuvent être réécrites dans le fichier de configuration du site

Sécuriser Apache

Fichier .htaccess

- Ces fichiers peuvent être utiles pour déléguer des possibilités de modification aux utilisateurs, par exemple les hébergements mutualisés

- Ils sont souvent utilisés dans les CMS lors de l’installation de modules qui génèrent automatiquement des entrées dans les fichiers .htaccess

Sécuriser Apache

Fichier .htaccess

- La directive **`AllowOverride`** permet de définir les instructions autorisées dans les fichiers .htaccess
- Interdire l'utilisation des fichiers .htaccess : **`AllowOverride None`**
- Autoriser les fichiers .htaccess : **`AllowOverride All`**
- N'autoriser que certains groupes d'instructions : **`AllowOverride Indexes AuthConfig`**

Restrictions d’accès

- Les directives **`order`**, **`allow`**, **`deny`**, **`Require`**, **`RequireAny`**, **`RequireNone`** et **`RequireAll`** permettent de définir des restrictions d’accès
- Depuis Apache 2.4, les directives **`order`**, **`allow`** et **`deny`** sont **dépréciées** mais toujours prises en compte
- Les nouvelles directives sont accessible avec le module **authz_host**, les anciennes avec le module **access_compat**

Sécuriser Apache

Besoin	Apache 2.4	Apache 2.2
Accès interdit pour tous	`Require all denied`	`Order deny, allow` `Deny from all`
Accès autorisé pour tous	`Require all granted`	`Order allow, deny` `Allow from all`

Accès autorisé uniquement pour les hôtes du réseau 10.21.0.0/16	`Require ip 10.21`	`Order deny, allow`
		`Deny from all`
		`Allow from 10.21.0.0/16`

Accès autorisé pour tous sauf les hôtes du réseau 10.21.0.0./16	`Require not ip 10.21`	`Order allow, deny`
		`Allow from all`
		`Deny from 10.21.0.0/16`

Sécuriser Apache

Règles multiples

Besoin	Apache 2.4

Accès autorisé pour les membres du groupe users se connectant depuis le réseau 10.21.0.0./16	`<RequireAll>`
	`  Require ip 10.21`
	`  Require group users`
	`</RequireAll>`

Autoriser les hôtes du réseau 10.21.0.0/16 ou l'utilisateur Gilles	`<RequireAny>`
	`  Require ip 10.21`
	`  Require user Gilles`
	`</RequireAny>`

Appliquer des restrictions d’accès

Démonstration

Sécuriser Apache

Chiffrer avec SSL/TLS

- Le protocole TLS (*Transport Layer Security*) a remplacé le protocole SSL (*Secure Socket Layer*)
- SSL fut créé par **Netscape** en 1994 et standardisé par l'IETF à partir de 1999, le renommant TLS
- Par habitude de langage, on parle de protocole SSL
- Protocole s'appliquant sur divers protocoles réseau autres que HTTP : FTP, IMAP, SMTP...

Sécuriser Apache

Chiffrer avec SSL/TLS

- Trois fonctions principales :
  - **Authentification du serveur** grâce à un certificat numérique
  - **Confidentialité des données** en les chiffrant sur le média
  - **Intégrité des données** grâce à la fonction de hachage
- Le module utilisé par Apache est **mod_ssl**

Sécuriser Apache

Certificats SSL

- On peut classer les certificats SSL en trois catégories :
  - **Certificats SSL payants** : à acheter chez des fournisseurs de certificats (Thawte, Verisign...)
  - **Certificats SSL autosignés** : autorité de certification et certificat créés par l'administrateur
    Attention, non reconnus par les navigateurs
  - **Certificats SSL Let's Encrypt** : nouvelle autorité de certification créée en 2015, permet de créer des certificats SSL gratuits, reconnus, générés par des scripts
    Fin 2017, plus de 46 millions de certificats générés

Sécuriser Apache

Créer un certificat autosigné

- Pour installer un certificat sur un site, il faut le certificat SSL et la clé privée

- Trois étapes :

1. Créer une clé privée
2. Créer un fichier de demande de signature de certificat (CSR)
3. Créer le certificat autosigné

- Les commandes suivantes permettent de générer un certificat SSL pour le site **www.sirius.com**

Sécuriser Apache

Créer un clé privée

- La clé privée est utilisée pour générer la demande mais aussi pour utiliser le certificat. **Ne surtout pas la perdre**

```
# openssl genrsa -des3 –out \ /etc/ssl/private/www.sirius.com.key
2048
```

- **genrsa** : génération de la clé privée avec l'algorithme RSA
- **-des3** : oblige l'utilisation d'un mot de passe sur la clé
- **-out** : nom du fichier de clé
- **2048** : taille de la clé

Sécuriser Apache

Créer le fichier CSR

- Ce fichier va être utilisé pour générer le certificat SSL soit par une autorité externe, soit pour un certificat autosigné

```
# openssl req -new
-key /etc/ssl/private/www.sirius.com.key
-out /etc/ssl/private/www.sirius.com.csr
```

- Attention aux informations demandées, ce sont les informations qui apparaîtront dans le certificat

- Le **Common Name** désigne l'URL du site

Sécuriser Apache

Créer le certificat

- Dernière étape, la création du certificat SSL :

```
# openssl x509 -req -days 90 \
-in /etc/ssl/private/www.sirius.com.csr \
-signkey /etc/ssl/private/www.sirius.com.key \
-out /etc/ssl/private/www.sirius.com.cert
```

- **`x509`** : le type de certificat X.509
- **`-days 90`** : certificat valide pendant 90 jours
- **`-in`** : en entrée, le fichier CSR
- **`-signkey`** : la clé privée

Mot de passe de la clé

- Lors de la création de la clé, il a été mis un mot de passe
- Si on laisse tel quel, à chaque démarrage d'Apache, il faudra saisir le mot de passe

```
# openssl rsa -in /etc/ssl/private/www.sirius.com.key
-out /etc/ssl/private/www.sirius.com.key.pem
```

Sécuriser Apache

Activer le module SSL

- Pour qu'Apache écoute sur le port HTTPS (443), il faut activer le module SSL :

```
# a2enmod ssl
```

- Ce qui va activer la configuration dans **`ports.conf`** :

```
<IfModule mod_ssl.c>
    Listen 443
</IfModule>
```

Sécuriser Apache

Hôte virtuel SSL

- Reste à créer un hôte virtuel sur le port 443, configuration minimale :

```
<VirtualHost *:443>
   ServerName www.sirius.com
   DocumentRoot /var/www/site

   SSLEngine On
   SSLCertificateKeyFile /etc/ssl/private/www.sirius.com.key.pem
   SSLCertificateFile /etc/ssl/private/www.sirius.com.cert
</VirtualHost>
```

Hôte virtuel SSL

- **`SSLEngine On`** : active le SSL pour cet hôte virtuel
- **`SSLCertificateKeyFile`** : chemin complet vers la clé
- **`SSLCertificateFile`** : chemin complet vers le certificat
- Le site suivant permet de générer une configuration complète : https://mozilla.github.io/server-side-tls/ssl-config-generator/

Créer un certificat autosigné

Démonstration

Sécuriser Apache

En-têtes

- Il est possible de cacher certaines informations envoyées par Apache dans les en-têtes et les messages d'erreur
- Dans **`/etc/apache2/conf-available/security.conf`** :
  - Cache la version d'Apache dans les en-têtes **`ServerTokens Prod`**
  - Cache la signature du serveur dans les pages d'erreur **`ServerSignature Off`**

Modifier des en-têtes

Démonstration

Conclusion

- Vous savez comment restreindre les accès aux répertoires ou aux URL
- Vous savez configurer un certificat SSL sur les sites