

**SECURINETS**



**CLUB DE LA SECURITE INFORMATIQUE  
INSAT**

# SECURILIGHT 2011

---

[Sécuriser un serveur web apache]

**Chef Atelier : Ben Mansour Nejib (RT5)**

**Zahar Sabrine (MPI)**

**Makni Emna (MPI)**

**Ben Hammouda Naim (RT5)**

**Dridi Marwa (RT5)**

**Hichri Feker (GL3)**

**30/11/2011**

## Présentation de l'atelier :

Le but de notre atelier est de :

Se familiariser avec le serveur web Apache2 sous Linux.

Comprendre le rôle des fichiers .htaccess et .htpasswd

Savoir les notions de base pour la sécurité d'un serveur web apache

**Note:** Nous avons travaillé cet atelier sous la version Ubuntu 10.04 et avec le serveur apache 2.2.14.

### I. Serveur web :

#### a. Définition et rôle :

Un serveur web est un logiciel permettant à des clients d'accéder à des pages web, c'est-à-dire en réalité des fichiers au format HTML à partir d'un navigateur.

Un serveur web est donc un « simple » logiciel capable d'interpréter les requêtes HTTP arrivant sur le port associé au protocole HTTP (par défaut le port 80), et de fournir une réponse avec ce même protocole.

Les principaux serveurs web sur le marché sont entre autres :

- Apache
- Microsoft IIS (Internet Information Server)
- Microsoft PWS (Personal Web Server)
- Xitami

#### b. Modules d'apache :

Apache contient plusieurs modules permettant de faciliter son fonctionnement, parmi ces modules on trouve :

- Modsecurity: Modsecurity est un pare-feu applicatif, dont le rôle est de filtrer les requêtes entrant sur un serveur HTTP Apache. Il se présente sous la forme d'un module apache, qui analyse les requêtes reçues grâce à l'emploi

d'une base des règles de requêtes considérées comme non souhaitées. Ces règles sont codées sous forme d'expressions régulières.

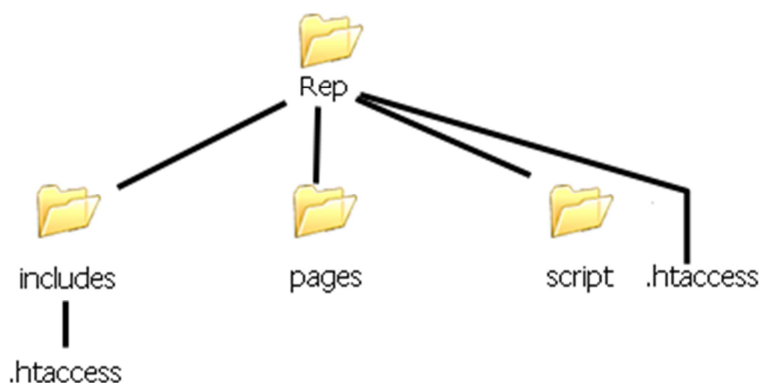
- Mod\_Ssl : Les services Web demandent souvent une authentification de la part des utilisateurs. Dans ce cas, ce dernier va fournir un couple login/mot de passe qui devra transiter sur le réseau entre le client et le serveur. Pour que ces identifiants ne circulent pas en clair sur le réseau, Apache offre la possibilité de chiffrer les échanges à l'aide d'OpenSSL.

### c. Les fichiers .htaccess :

Un fichier .htaccess est un fichier de configuration pour Apache qui sert à :

- Protéger un répertoire ou un fichier par mot de passe.
- Modifier les droits d'accès.
- Créer des redirections.
- La gestion des pages d'erreurs.

#### Exemple :



On peut définir plusieurs fichiers .htaccess dans un même répertoire.

Les règles de fichier .htaccess s'appliquent au répertoire contenant ce fichier et ses sous-répertoires

## II. Installation d'apache 2 :

-Pour installer apache on utilise la commande apt-get install

**Sudo apt-get install apache2**

-Le répertoire racine du serveur :

**/var/www/**

-Le répertoire contenant les fichiers de configuration d'apache :

**/etc/apache2**

-Les fichiers log d'apache :

**/var/log/apache2**

### Remarque :

Ces fichiers sont très importants et très utiles puisque l'administrateur peut contrôler et surveiller toutes les actions effectués sur le serveur et analyser les messages d'erreur ainsi il peut détecter à partir de ces fichiers les problèmes et les tentatives d'attaques.

Voici les principaux répertoires à connaître pour configurer le serveur Web sous Debian :

- **/etc/apache2/apache2.conf** : fichier principal de configuration d'Apache 2.
- **/etc/apache2/ports.conf** : fichier contenant les ports sur lesquels Apache doit écouter.
- **/etc/apache2/sites-available/** : dossier contenant les définitions des différents vhosts.
- **/etc/apache2/sites-enable/** : dossier contenant les vhosts actifs.
- **/etc/apache2/mods-available/** : dossier contenant les différents modules installés.
- **/etc/apache2/mods-enable/** : dossier contenant les modules actifs.

– `/etc/apache2/ssl/` : fichiers relatifs à la configuration du `mod_ssl`.

### III. Sécuriser l'accès par login et mot de passe :

#### a. Pour un site :

Créer le fichier `.htaccess`

```
AuthUserFile "/home/securinets/.htpasswd"  
AuthName "Access Admin"  
AuthType Basic  
<Limit GET POST>  
require valid-user  
</Limit>
```

Créer le fichier `.htpasswd` avec le chemin indiqué dans `.htaccess` et ajouter les utilisateurs et leurs mots de passe avec la commande

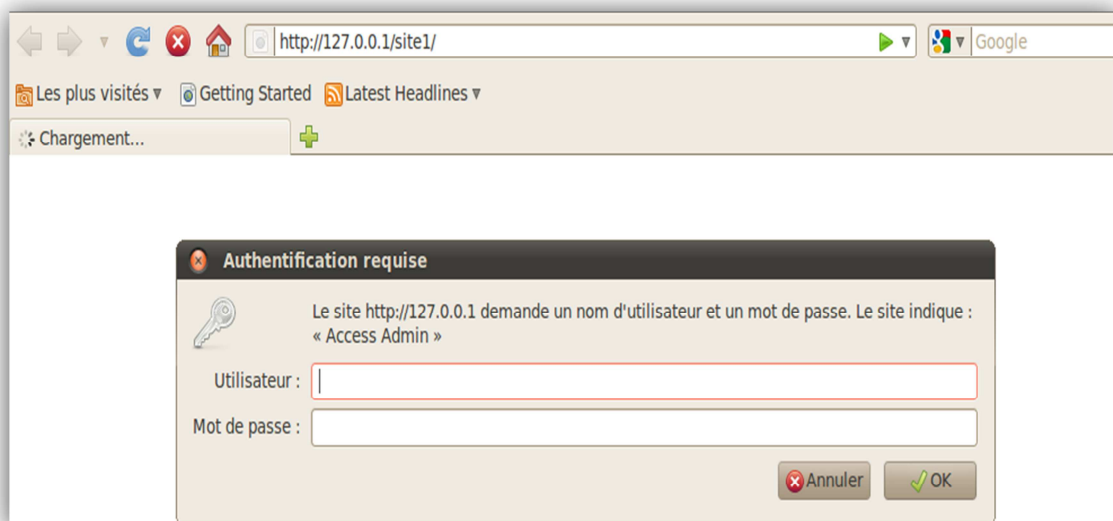
**`htpasswd -c /var/www/.htpasswd nom_utilisateur`**

Cette commande permet de créer un fichier contenant le couple login et mot de passe avec le cryptage de mot de passe.

Voici un exemple de contenu de ce fichier :

```
atelier:zcQS6Ts.aE2ks
```

Modifier le `AllowOverride` dans le fichier `/etc/apache2/sites-available/default` en la mettant à **all** pour `<Directory />` et `<Directory /var/www>`



## b. Pour un fichier dans le site :

Il faut juste modifier le fichier .htaccess de cette manière :

```
<Files tichier_avec_pass>
AuthUserFile "/home/site1/.htpasswd"
AuthName "You need a pass to read file"
AuthType Basic
<Limit GET POST>
require valid-user
</Limit>
</Files>
```

## IV. Cacher l'identité du serveur apache :

### a. Pour une fausse adresse :

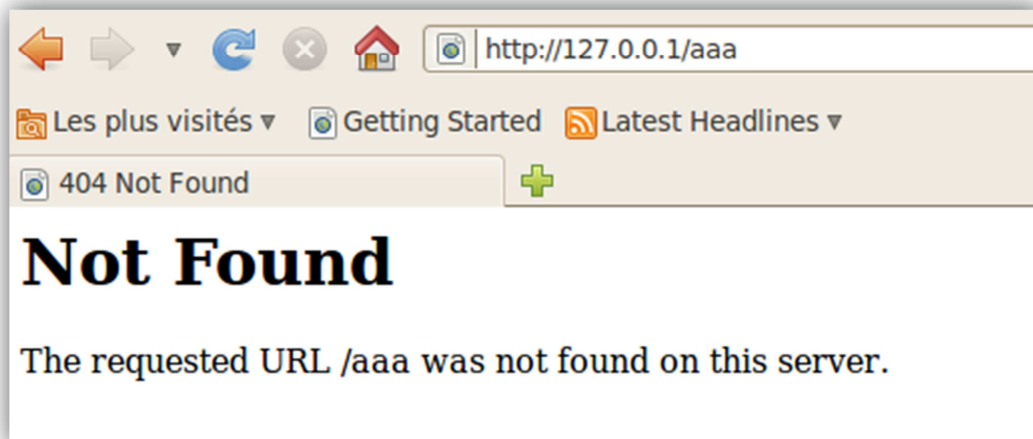
Lorsque l'utilisateur donne une fausse adresse, le serveur rend un message d'erreur contenant sa version et le système sur lequel il fonctionne. Nous allons cacher ses informations supplémentaires.

Pour cela il faut agir sur le fichier /etc/apache2/conf.d/security et mettre ServerSignature à off

Le message d'erreur habituel :



Après la modification :



**b. Pour un accès correct à la page :**

Un utilisateur malicieux peut accéder à la page et effectuer une analyse des requêtes pour en sortir des informations qui peuvent constituer un danger sur notre serveur web.

Pour remédier à ce problème on modifie dans le fichier `/etc/apache2/conf.d/security` et on met `ServerTokens` à `Prod`

`Prod` : signifie qu'on va afficher seulement le mot `apache` dans la requête.

Voici une analyse de requête avec `TamperData` qui montre la version d'apache et le système d'exploitation utilisé :

**Tamper Data - Requêtes en cours**

Démarrer altération Arrêter altération Effacer Options Aide

Filtre  Afficher tout

Temps	Durée	Durée totale	Taille	Méthode	État	Type de contenu	URL	Indicateurs de chargement
17:21:01.312	74 ms	74 ms	290	GET	401	text/html	http://19...	LOAD_DOCUMENT_URI LOAD_INITI...
17:21:36.683	42 ms	87 ms	-1	GET	304	application/x-unknown...	http://19...	LOAD_DOCUMENT_URI LOAD_INITI...
17:21:36.774	0 ms	0 ms	-1	GET	Charger d...	inconnu	http://19...	LOAD_FROM_CACHE VALIDATE_NE...
17:21:36.782	0 ms	0 ms	unknown	GET	pending	unknown	http://19...	LOAD_NORMAL

Nom de l'en-tête de la req...	Valeur de l'en-tête de la requête	Nom de l'en-tête de la réponse	Nom de l'en-tête de la réponse
Host	192.168.1.7	Status	Authorization Required - 401
User-Agent	Mozilla/5.0 (Windows NT 5.1; rv:6.0.2) Gecko/20100101 Firefox/3.5.1	Date	Fri, 18 Nov 2011 07:14:56 GMT
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8	Server	Apache/2.2.14 (Ubuntu)
Accept-Language	fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3	WWW-Authenticate	Basic realm="Access Admin"
Accept-Encoding	gzip, deflate	Vary	Accept-Encoding
Accept-Charset	ISO-8859-1,utf-8;q=0.7,*;q=0.7	Content-Encoding	gzip
Connection	keep-alive	Content-Length	290
If-Modified-Since	Sun, 13 Nov 2011 02:06:21 GMT	Keep-Alive	timeout=15, max=100
If-None-Match	"68dc2-cc-4b1943352cbf"	Connection	Keep-Alive
		Content-Type	text/html; charset=iso-8859-1

Maintenant après la modification on aura seulement le mot ‘apache’

**Tamper Data - Requêtes en cours**

Démarrer altération Arrêter altération Effacer Options Aide

Filtre  Afficher tout

Temps	Durée	Durée totale	Taille	Méthode	État	Type de contenu	URL	Indicateurs de chargement
17:24:57.088	81 ms	81 ms	290	GET	401	text/html	http://192...	LOAD_DOCUMENT_URI LOAD_INITI...
17:25:07.318	32 ms	65 ms	-1	GET	304	application/x-unknown...	http://192...	LOAD_DOCUMENT_URI LOAD_INITI...

Nom de l'en-tête de la req...	Valeur de l'en-tête de la requête	Nom de l'en-tête de la réponse	Nom de l'en-tête de la réponse
Host	192.168.1.7	Status	Authorization Required - 401
User-Agent	Mozilla/5.0 (Windows NT 5.1; rv:6.0.2) Gecko/20100101 Firefox/3.5.1	Date	Fri, 18 Nov 2011 07:18:51 GMT
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8	Server	Apache
Accept-Language	fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3	WWW-Authenticate	Basic realm="Access Admin"
Accept-Encoding	gzip, deflate	Vary	Accept-Encoding
Accept-Charset	ISO-8859-1,utf-8;q=0.7,*;q=0.7	Content-Encoding	gzip
Connection	keep-alive	Content-Length	290
If-Modified-Since	Sun, 13 Nov 2011 02:06:21 GMT	Keep-Alive	timeout=15, max=100
If-None-Match	"68dc2-cc-4b1943352cbf"	Connection	Keep-Alive
		Content-Type	text/html; charset=iso-8859-1

## V. Lutter contre les attaques DOS :

### a. Principe d'attaque DOS :

Une attaque DoS consiste à tenter de bloquer l'accès à un serveur en monopolisant toutes ses ressources.

Les ordinateurs ont une capacité de traitement limitée dans un temps donné. Si le nombre de requêtes excède la limite prévue, les nouvelles requêtes sont mises en attente. Si la file d'attente sature à son tour, les nouvelles requêtes sont ignorées et le service est interrompu momentanément.

Les attaques DoS les plus connues sont :

- ***SYN flood*** qui consiste à initier une transaction TCP avec un serveur sans l'informer qu'elle est établie. Le serveur réserve les ressources en attendant la réponse.
- ***Ping flood*** qui consiste à saturer la bande passante d'un serveur en l'inondant de ping.

Pour diminuer les attaques DOS on peut utiliser 3 méthodes :

### b. Modifier le fichier de configuration apache2.conf :

On peut agir sur le fichier de configuration d'apache en modifiant le nombre maximal de connexions simultanées, la durée d'une connexion, le timeout, le nombre maximal de demandes de connexions accepté, le temps d'attente pour une nouvelle demande par le même client et les informations concernant la taille d'entête d'une requête.

Certains réglages de la configuration d'Apache peuvent aussi minimiser les problèmes

- La directive RequestReadTimeout permet de limiter le temps que met le client pour envoyer sa requête.
- La valeur de la directive Timeout doit être diminuée sur les sites sujets aux attaques DoS. Une valeur de quelques secondes devrait convenir. Cependant, comme Timeout est actuellement concerné par de nombreuses opérations

différentes, lui attribuer une valeur trop faible peut provoquer des problèmes avec les scripts CGI qui présentent un long temps de réponse.

- La valeur de la directive `KeepAliveTimeout` doit aussi être diminuée sur les sites sujets aux attaques DoS. Certains sites désactivent même complètement le "maintien en vie" (`keepalives`) à l'aide de la directive `KeepAlive`, ce qui bien sûr présente des inconvénients en matière de performances.
- Les valeurs des différentes directives fournies par d'autres modules et en rapport avec des délais doivent aussi être vérifiées.
- Les directives `LimitRequestBody`, `LimitRequestFields`, `LimitRequestFieldSize`, `LimitRequestLine`, et `LimitXMLRequestBody` doivent être configurées avec prudence afin de limiter la consommation de ressources induite par les demandes des clients.

La modification de ces paramètres est faite selon le service proposé par notre serveur et les exigences des connexions, pour cette raison la modification de ce fichier est insuffisante pour limiter contre l'attaque DOS.

### c. Utiliser le module `mod_evasive` :

Le `mod_evasive` crée une table dynamique des IP clients et des URL demandées. Par défaut, il blacklist et renvoie une erreur 403 si l'IP :

- fait plus de X requêtes par seconde sur une uri
- fait plus de X requêtes par seconde sur un process Apache
- continue de faire des requêtes alors qu'il est blacklisté

Pour l'installer :

```
# apt-get install libapache2-mod-evasive
```

Pour l'activer :

```
# a2enmod mod-evasive
```

Pour définir la configuration :

```
# vi /etc/apache2/mods-available/mod-evasive.conf
```

Ajoutons et ajustons selon nos besoins les lignes suivantes :

```
# mod_evasive

<IfModule mod_evasive20.c>
    # taille de la table de surveillance, a augmenter pour
    les DDoS
    DOSHashTableSize 2048
    # Nb maximum de refresh d'une uri par periode
    DOSPageCount 5
    # Duree de la periode pour la uri
    DOSPageInterval 1
    # Nb maximum de requete sur le site par periode
    DOSSiteCount 150
    # Duree de la periode pour le site
    DOSSiteInterval 1
    # Duree du blacklisting de l'IP reconduite si
    retentative
    DOSBlockingPeriod 10
    # Execute une action quand une IP est bloquee
    DOSSystemCommand "/sbin/iptables -I INPUT -s %s -j
    DROP"

    # envoie de mail quant une IP est bloquee
    DOSEmailNotify "admin@localhost"
    # repertoire des logs
    DOSLogDir "/var/log/apache2/"
    # Whitelist non surveillee
    DOSWhiteList 127.0.0.1
    DOSWhitelist 192.168.1.*
</IfModule>
```

#### d. Utiliser le firewall iptables :

Contre les attaques DoS de haut niveau, l'usage d'un **firewall** est le plus efficace. Le programme **iptables** est beaucoup moins gourmand en CPU et en mémoire qu'un processus apache2 et repousse donc les limites de tolérance d'une attaque DoS pour une machine donnée.

Il permet également un réglage plus fin. Par exemple, iptables permet de limiter le nombre de connexions simultanées provenant d'une adresse IP ou d'un réseau, le

nombre de connexions depuis une source, le tout sur une période donnée.

Les règles suivantes permettent de limiter le nombre de connexion depuis une source à 50 hits pendant 10 secondes sur le port 80 :

```
# iptables -A INPUT -i eth0 -p tcp --dport 80 -m state --state NEW -m  
recent --set --name WEB  
# iptables -A INPUT -i eth0 -p tcp --dport 80 -m state --state NEW -m
```