Unité Réseaux du CNRS

Configuration du logiciel Apache
Aspects sécurité

Claude Gross
6 février 1997

Table des matières

Introduction

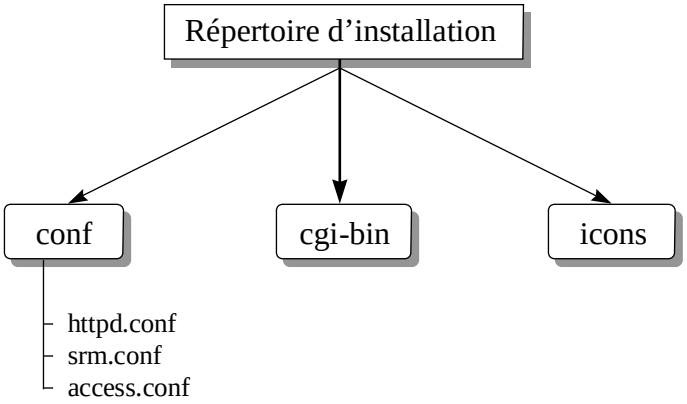
Le logiciel Apache est actuellement le logiciel serveur http le plus utilisé dans l'Internet. Doté de nombreuses fonctionnalités, performant et gratuit, il constitue un choix très intéressant pour ceux voulant mettre en place un service WWW.

Mais comme pour tout logiciel, le fait d'offrir de nombreuses fonctionnalités implique également une complexité plus grande d'utilisation et en particulier de configuration. Cela entraîne également très souvent, dans le domaine de l'Internet, des problèmes potentiels supplémentaires concernant la sécurité.

Ce papier essaie de faire le tour de la question concernant les aspects de la configuration d'Apache qui entrent en jeux dans les problèmes de sécurité. Il n'aborde donc pas tous les problèmes de configuration d'Apache mais se consacre uniquement à ceux liés à la sécurité.
Les conseils qui y sont donnés peuvent être adaptés, car l'administration des serveurs http varie beaucoup d'un site à un autre.

Généralités

L'installation du logiciel Apache se fait, par défaut, dans le répertoire /usr/local/etc/httpd. Ce répertoire contient en particulier un répertoire conf qui va contenir les fichiers de configuration d'Apache : httpd.conf, srm.conf et access.conf



Ces fichiers sont organisés de la façon suivante :

- httpd.conf contient les directives de configuration générale
- srm.conf contient les directives concernant les ressources du serveur
- access.conf contient les directives concernant la politique d'accès au serveur.

A coté de ces 3 fichiers, on peut également utiliser des fichiers de configuration que l'on place dans les répertoires mêmes des documents du serveur. Ces fichiers ont un nom particulier (par défaut .htaccess) et peuvent contenir à peu près les mêmes directives que les 3 fichiers ci-dessus.

Les aspects sécurité du logiciel Apache ne concernent pas uniquement le fichier access.conf. Elles sont liés à l'utilisation d'un ensemble de directives qui se trouvent dans les 3 fichiers de configuration et à l'utilisation de certaines fonctionnalités du logiciel :

- politique d'accès aux documents : accès libre, filtrage par rapport aux domaines ou accès par utilisateur et mot de passe.
- utilisation des programmes CGI
- utilisation des directives « Server Side Include ».
- accès aux répertoires des utilisateurs de la machine où tourne le serveur Apache

Configuration générale

Lancement du daemon

Le premier point sensible concernant la sécurité des serveurs httpd est la façon dont ceux-ci sont démarrés et en particulier l'identité sous laquelle est démarré le serveur. On ne traite ici que le cas où la directive ServerType a pour valeur *standalone*, c'est à dire où on ne démarre pas le serveur par inetd, ce qui n'est pratiquement jamais le cas, pour des raisons de performance.

Si le serveur est démarré par un utilisateur autre que root, tous les processus appartiendront à cet utilisateur. S'il est démarré par root, le processus père appartiendra à root, mais tous les processus fils, qui répondront aux requêtes, appartiendront à l'utilisateur définit par les directives User et Group dans le fichier httpd.conf.

En cas de trou de sécurité, et donc de possibilité d'intrusion sur la machine, à travers le serveur http, ce problème pourra être exploité avec les droits de l'utilisateur auquel appartient les processus fils. Il est donc fort déconseillé de donner la valeur *root* à la directive User, mais de choisir plutôt un utilisateur de la machine sans droits particuliers, par exemple nobody ou un utilisateur créé spécialement pour cela.

Répertoire racine du serveur

Il est définit par la directive DocumentRoot dans le fichier srm.conf. Il représente en gros la partie de l'espace disque de la machine qui sera accessible via le serveur http, c'est à dire l'endroit ou seront déposé les fichiers HTML que l'on veut diffuser.

Il est évident que la valeur de cette directive ne devra pas permettre de rendre accessibles des fichiers que l'on ne souhaite pas rendre accessibles, en particulier les fichiers du système.

Exemples :

Valeur raisonnable :

```
DocumentRoot /usr/local/etc/httpd/htdocs
```

Valeur non raisonnable :

```
DocumentRoot /
```

Accès aux documents

Mettre en place un service WWW ne signifie pas pour autant rendre accessible à tout le monde les documents du serveur. Certains services WWW ne sont d'ailleurs accessibles qu'à une certaine population. Cela signifie qu'il faut définir une politique d 'accès au service, décider qui a accès à quoi, et configurer le logiciel de manière à appliquer cette politique.
Le logiciel Apache comme la plupart des autres logiciels serveurs http permet 2 type de protection :

- une protection par domaine, qui permet de définir des droits d'accès en fonction des noms de machines ou de domaines
- une protection par utilisateur, qui permet de protéger tout ou partie du serveur par nom d'utilisateur et mot de passe (4).

La première méthode est simple à mettre en œuvre et ne nécessite pratiquement pas d'administration particulière. Par contre, la seconde implique la gestion de comptes utilisateur et donc plus de travail.

Les deux méthodes se définissent dans le fichier `access.conf`. Celui-ci contient au moins une directive `<Directory>` qui va définir la politique par défaut pour tous les documents du serveur. On peut ensuite ajouter des directives `<Directory>` pour modifier les caractéristiques de certaines sous-arborescences.

La directive `<Directory>` est un bloc pouvant contenir un certain nombre de sous-directives :

- `Options` : est suivi par une liste d'options possibles :
  - `Indexes` : indique que l'on peut avoir accès à la liste des fichiers des répertoires.
  - `Includes` : indique que l'on peut avoir des fichiers contenant des directives « Server Side Include » (SSI) dans cette arborescence.
  - `includesNOEXEC` : même chose que `Includes` mais on interdit la commande `#exec` ainsi que l'inclusion de script CGI
  - `FollowSymLinks` : on autorise l'accès aux liens symboliques
  - `SymLinksIfOwnerMatch` : on autorise l'accès aux liens symboliques si le propriétaire est le même aux 2 extrémités du lien.
  - `ExecCGI` : on autorise des programmes CGI dans cette arborescence.
- `AllowOverride` : indique si on peut ou non utiliser des fichiers de configuration à l'intérieur des répertoires. Ces fichiers, appelés par défaut `.htaccess`, peuvent

contenir à peu près les mêmes directives que les fichier `httpd.conf`, `srm.conf` et `access.conf`, et ne concernent que le répertoire dans lequel ils se trouvent. Valeurs possibles :

- `All` : les fichiers .htaccess sont autorisés
- `None` : les fichiers .htaccess sont interdits

On peut également utiliser l'une ou l'autre des valeurs suivantes :

- `AuthConfig` : autorise les directives d'autorisation (`AuthDBMGroupFile`, `AuthDBMUserFile`, `AuthGroupFile`, `AuthName`, `AuthType`, `AuthUserFile`, `require`, etc.).
- `FileInfo` : autorise les directives contrôlant le type des documents (`AddEncoding`, `AddType`, `DefaultType`, `ErrorDocument`, `LanguagePriority`, etc.).
- `Indexes` : autorise les directives concernant la présentation des répertoires (`AddDescription`, `AddIconByEncoding`, `AddIconByType`, `DefaultIcon`, `DirectoryIndex`, `FancyIndexing`, `HeaderName`, `IndexIgnore`, `IndexOptions`, `ReadmeName`, etc.).
- `Limit` : autorise les sous-directives de la directive `Limit` (`allow`, `deny` and `order`).
- `Options` : autorise les directives `Options` et `XBitHack`

- `<Limit>` : est un bloc contenant des sous-directives permettant de définir les droits d'accès associés à une ou plusieurs méthodes d'accès (`GET`, `POST`...) :
  - `order` : indique l'ordre dans lequel on va définir les droits :
    - `order allow deny`

    ou

    - `order deny allow`
  - `allow` : autorise un ou plusieurs domaines
  - `deny` : interdit un ou plusieurs domaines
  - `require` : dans le cas d'accès par utilisateur et mot de passe, indique le ou les groupes ou le ou les utilisateurs ayant accès.

Exemple :

```
<Directory /usr/local/etc/httpd/htdocs>
    Options Indexes SymLinksIfOwnerMatch Includes
    AllowOverride None
    <Limit GET>
       order allow,deny
       allow from all
    </Limit>
</Directory>
<Directory /usr/local/etc/httpd/htdocs/docs>
    Options +ExecCGI
    AllowOverride None
    <Limit GET>
       order allow,deny
       allow from all
    </Limit>
</Directory>
```

On suppose ici que /usr/local/etc/httpd/htdocs est le répertoire racine du serveur. Dans l'exemple ci-dessus, la première directive <Directory> permet de définir la politique générale d'accès au serveur. La suivante spécifie les mêmes propriétés pour l'arborescence /usr/local/etc/httpd/htdocs/docs mais ajoute la possibilité d'y mettre des programmes CGI (+ExecCGI)

La protection par domaines

Elle consiste à donner, ou refuser, l'accès de certains documents en fonction du domaine auquel appartient la machine à partir de laquelle est faite la requête.

Exemple :

```
<Directory /usr/local/etc/httpd/htdocs/>
    Options Indexes SymLinksIfOwnerMatch Includes
    AllowOverride None
    <Limit GET>
      order allow,deny
      allow from all
    </Limit>
</Directory>
<Directory /usr/local/etc/httpd/htdocs/local>
    <Limit GET>
       order deny,allow
      deny from all
      allow from .urec.fr
    </Limit>
</Directory>
```

Dans cette exemple, la première directive <Directory> indique que les documents du serveur sont accessibles à tout le monde.
La deuxième directive <Directory> définit un filtre pour les documents dans /usr/local/etc/httpd/htdocs/local qui ne sont accessibles qu'aux machines appartenant au domaine urec.fr.

La protection par utilisateurs

Pour mettre en place ce type de protection, il faut procéder de la manière suivante :

- récupérer ou compiler le programme htpasswd dont les sources sont fournies avec Apache
- créer avec cette commande un fichier (par exemple htpasswd) contenant les utilisateurs ainsi que leur mot de passe. Ce fichier a, à peu près, la même syntaxe que le fichier passwd sur les systèmes UNIX.
- créer à la main un fichier contenant les groupes d'utilisateurs, par exemple htgroup.
- Syntaxe : groupe : user1 user2 user3...
- mettre à jour le fichier access.conf

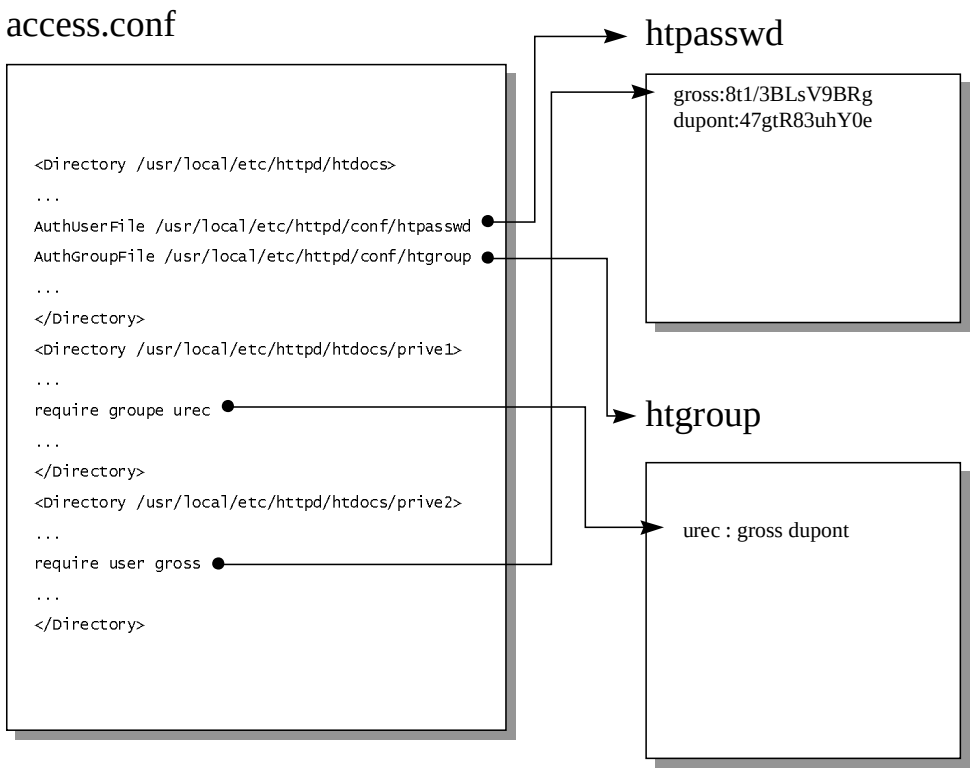
Exemple :

```
<Directory /usr/local/etc/httpd/htdocs/>
    Options Indexes SymLinksIfOwnerMatch Includes
    AllowOverride None
    AuthType Basic
    AuthUserFile /usr/local/etc/httpd/conf/htpasswd
    AuthGroupFile /usr/local/etc/httpd/conf/htgroup
    <Limit GET>
       order allow,deny
       allow from all
    </Limit>
</Directory>
<Directory /usr/local/etc/httpd/htdocs/prive1>
    AuthName Groupe Urec
    <Limit GET POST>
        require group urec
    </Limit>
</Directory>
<Directory /usr/local/etc/httpd/htdocs/prive2>
    AuthName Prive2
    <Limit GET POST>
        require user gross
    </Limit>
</Directory>
```

Dans cette exemple, on crée, pour tous les fichiers sous les arborescences /usr/local/etc/httpd/htdocs/prive1 et /usr/local/etc/httpd/htdocs/prive2, une protection par utilisateur. Dans le premier cas, l'accès est réservé aux utilisateurs appartenant au groupe urec, dans le deuxième, seul l'utilisateur gross aura droit d'accès.

- AuthType : indique le type d'authentification
- AuthUserFile : indique le nom du fichier utilisateurs
- AuthGroupFile : indique le nom du fichier groupe
- AuthName : indique une chaîne de caractère qui sera utilisé dans la fenêtre du navigateur qui demandera le nom d'utilisateur et le mot de passe.
- require : indique les utilisateurs ou les groupes d'utilisateurs qui ont le droit d'accès.

La principale difficulté dans ce type de configuration est de ne pas se tromper dans les noms de fichiers, de groupes ou d'utilisateurs. Les relations peuvent se représenter ainsi :



Programmes CGI

Les programmes CGI sont des programmes destinés à être exécutés par le serveur http, en général pour le traitement d'un formulaire, ou dans un autre but. Ces programmes, écrits dans n'importe quel langage supporté sur la machine, sont exécutés avec les droits de l'utilisateur auquel appartient les processus fils du serveur (utilisateur qui a lancé le serveur si ce n'est pas root, ou utilisateur identifié par les directives User et Group dans httpd.conf dans le cas d'un lancement par root).

L'utilisation de programmes CGI induit des problèmes de sécurité potentiels supplémentaires car ils peuvent eux mêmes contenir des trous de sécurité. Ainsi des programmes CGI peuvent, accidentellement ou de façon malveillante, annihiler complètement les efforts faits pour rendre sécurisé le logiciel Apache lui-même (7).

Pour utiliser des programmes CGI, il existe différentes possibilités au niveau de la configuration d'Apache :

- la directive ScriptAlias dans srm.Conf permet de définir un ou plusieurs répertoires qui seront connus du serveur comme contenant des programmes CGI. Toute requête à un fichier de ces répertoires sera interprétée par le serveur comme une demande d'exécution du fichier demandé.

  Exemple :

  ```
  ScriptAlias /cgi-bin/ /usr/local/etc/httpd/cgi-bin/
  ```

- si la directive Options pour un répertoire dans access.conf contient la valeur ExecCGI, cela implique que ce répertoire peut contenir des programmes CGI. Cela permet d'avoir ainsi des programmes CGI ailleurs que dans les répertoires définis par ScriptAlias. Cette méthode implique également que le serveur puisse distinguer les fichiers à exécuter des autres fichiers. Cela est fait à l'aide de la directive AddHandler dans srm.conf qui permet d'associer un suffixe particulier aux fichiers contenant les programmes CGI.

  Exemple :

  ```
  AddHandler cgi-script .cgi
  ```

Pour essayer de limiter les risques induits par l'utilisation de programmes CGI, on peut :

- ne pas utiliser de programmes CGI. C'est la méthode la plus simple et la plus efficace.
- limiter les programmes CGI dans un répertoire précis, ce qui permet à l'administrateur du serveur de contrôler les programmes CGI existants sur le serveur

Si l'administrateur du serveur est le seul à y déposer des fichiers, il peut évidemment s'autoriser à mettre des programmes CGI où il veut. Si ce n'est pas le cas, soit il interdira les CGI dans les répertoires qu'il ne contrôle pas, soit il devra avoir une confiance complète envers les autres personnes déposant des fichiers sur le serveur.

Remarque :
Les logiciels serveurs http comme Apache sont souvent fournis avec un certain nombre d'exemples de programmes CGI qui sont, de plus, dans le répertoire définis par défaut avec `ScriptAlias`. Il est fortement recommandé de ne pas laisser accessibles ces exemples et de supprimer tout programme CGI qui n'a pas d'utilité particulière dans le serveur (6).

Les directives « Server Side Include »

Les directives « Server Side Include » (SSI) ont été introduits par les développeurs du logiciel serveur http du NCSA duquel est issu Apache. Ces directives sont maintenant supportées par d'autres logiciels serveur http comme celui de Netscape par exemple.

Les SSI sont la possibilité d'introduire dans les fichiers HTML des directives qui seront traitées par le logiciel serveur, lors d'une requête à ces fichiers, avant leur transfert au client. Pour activer les SSI dans Apache, il faut utiliser la directive AddHandler dans le fichier srm.conf de la manière suivante :

```
AddHandler server-parsed <suffixe>
```

ou <suffixe> est le suffixe que l'on veut associer aux fichiers contenant des SSI, par exemple « .shtml ».

Exemples :

- <!--#include virtual="/divers/intro.html" -->

  le serveur remplacera cette directive par le contenu du fichier /divers/intro.html qui est donné par son nom sur le serveur

- <!--#include file="intro.html" -->

  dans ce cas le nom du fichier est relatif au répertoire courant. Dans ces 2 cas, on ne peut normalement accéder qu'aux fichiers accessibles par ailleurs par le serveur.

- <!--#exec cmd="/usr/bin/date" -->

  le serveur remplace cette directive par le résultat de la commande « /usr/bin/date ». Toute commande du système peut être ainsi utilisée.

- <!--#exec cgi="/cgi-bin/date" -->

  dans ce cas la commande à exécuter est un programme CGI dont le chemin est relatif au serveur.

- <!--#echo var="LAST_MODIFIED" -->

  le serveur remplace cette directive par la valeur de la variable d'environnement « LAST_MODIFIED ».

Le problème principal avec les SSI est donc la possibilité de faire exécuter des programmes par le serveur. En cela, le problème est sensiblement le même qu'avec les programmes CGI. Pour réduire les risques liés à l'utilisation des SSI on peut :

- ne pas utiliser les SSI en ne mettant pour aucun des répertoires dans `access.conf` la valeur `Includes` dans la directive `Options`. C'est la méthode la plus simple et la plus efficace.
- utiliser la valeur `IncludesNOEXEC` pour Options qui permet l'inclusion de fichiers mais interdit l'utilisation de `#exec` et l'inclusion de programmes CGI.
- si vous mettez la valeur `Includes` dans la directive Options, il vous faut alors contrôler sérieusement le contenu de vos fichiers, surtout si vous laissez à d'autres personnes la possibilité de déposer des fichiers sur votre serveur.

Les documents des utilisateurs

Le logiciel Apache comme d'autres logiciels serveurs http permet de rendre accessibles des fichiers appartenant au répertoire personnel des utilisateurs de la machine. Pour cela il faut configurer Apache de la manière suivante :

- dans le fichier srm.conf, utiliser la directive UserDir de la manière suivante :

  UserDir <dir>

  ou <dir> est le nom du répertoire que les utilisateurs devront créer dans leur répertoire personnel. Tous les fichiers contenus dans ces répertoires seront alors accessibles avec des URLs de la forme :

  http ://machine.domaine/~dupont/a/b/c.html

  le fichier c.html a comme nom complet sur la machine :

  <homedir de dupont>/< UserDir>/a/b/c.html

Le fait de rendre ainsi accessibles des fichiers des utilisateurs oblige à prendre certaines précautions :

- créer dans le fichier access.conf des directives <Directory> spécifique pour les répertoires contenant des répertoires personnels d'utilisateurs.
- pour ces directives <Directory> :
  - interdire l'utilisation de fichiers .htaccess qui pourrait modifier la configuration que vous avez mis en place :

    AllowOverride None

  - interdire l'utilisation de programmes CGI ⇒ pas de valeur ExecCGI dans Options
  - interdire l'utilisation des SSI, ou au moins de la commande #exec et de l'inclusion de CGI ⇒ pas de valeur Includes ou valeur IncludesNOEXEC dans Options.

Exemple

<Directory /home>
  Options Indexes
  AllowOverride None
  <Limit GET POST>
  allow from all
  </Limit>
</Directory>
	répertoire contenant les répertoires des utilisateurs. On met le minimum de possibilités au niveau de ce répertoire.

<Directory /usr/local/etc/httpd/htdocs>.
  Options Indexes SymLinksIfOwnerMatch
  AllowOverride All
  <Limit GET>
  order allow,deny
  allow from all
  </Limit>
</Directory>
	répertoire racine.

<Directory /usr/local/etc/httpd/htdocs/local>
  <Limit GET>
  order deny,allow
  deny from all
  allow from .labo.fr
  </Limit>
</Directory>
	répertoire contenant des documents réservés au site.

<Directory /usr/local/etc/httpd/htdocs/prive>
  Options +Includes +ExecCGI
  AllowOverride All
  AuthType Basic
  AuthUserFile /usr/local/etc/httpd/conf/htpasswd
  AuthGroupFile /usr/local/etc/httpd/conf/htgroup
  AuthName Acces reserve
  <Limit GET>
  require group labo
  </Limit>
</Directory>
	répertoire particulier dont l'accès est protégé par utilisateur et mot de passe. On suppose cette partie administrée de façon plus stricte et donc on peut ajouter des options.

Conclusion

La sécurisation d'un serveur http utilisant le logiciel Apache doit se faire à différents niveaux. Si l'administrateur est seul maître du contenu du serveur, il est clair que les choses seront plus simples, mais la plupart du temps ce n'est pas le cas et certaines précautions sont à mettre en œuvre. En particulier :

- ➔ n'oubliez pas que la protection de documents par utilisateur et mot de passe, même si elle offre une protection correcte, n'est pas une garantie absolue. Les mots de passe transitant en clair sur le réseau, il est toujours possible d'être attaqué ainsi.
- ➔ désactivez les « Server Side Include » partout où c'est possible.
- ➔ interdisez les CGI là où ils ne sont pas nécessaires. De plus, prenez grand soin dans l'écriture des programmes CGI. Il est très facile d'introduire un trou de sécurité ainsi.
- ➔ protégez les répertoires des utilisateurs en y limitant au maximum les fonctionnalités : pas de SSI, pas de CGI, pas de fichiers .htaccess.
- ➔ configurez les répertoires avec «AllowOverride None » partout où c'est possible.

Références

1. Apache documentation
http://www.apache.org/docs

2. Apache Week
http://www.apacheweek.com/

3. Security tips for server configuration,
http://www.apache.org/docs/misc/security_tips.html

4. Using User Authentication
http://www.apacheweek.com/features/userauth

5. Making your setup more secure
http://hoohoo.ncsa.uiuc.edu/docs/tutorials/security.html

6. CERT advisorie, CA-96.06.cgi_example_code,
ftp://ftp.urec.fr/pub/securite/CERTs/miroir.cert.org/cert_advisories/CA-96.06.cgi_example_code

7. The World Wide Web Security FAQ
http://www-genome.wi.mit.edu/WWW/faqs/www-security-faq.html

8. Security Concerns on the Web
http://hoohoo.ncsa.uiuc.edu/security/

Voir aussi les recommendations du comité de cordination des serveurs web du CNRS : http://www.cnrs.fr/Gazette/Comite/comite.html