

Sécurisation

Serveur Web Apache

06/2011

Version 1.0
Release 06/2011

Ce document est une traduction de « *Guide to the Secure Configuration of Red Hat Enterprise Linux 5* » produit par la NSA (National Security Agency) et disponible à l'adresse suivante :

http://www.nsa.gov/ia_files/os/redhat/rhel5-guide-i731.pdf

Cette traduction concerne la partie 3.16.3 « *Secure the Apache Configuration* » à la partie 3.16.5.3 « *Run Apache in a chroot Jail if Possible* »

Licence du document	
<p>Le document original est la propriété de la NSA, L'agence pour la Sécurité Nationale américaine.</p> <p>http://www.nsa.gov</p> <p>La traduction et les modifications sont sous licence Creative Commons</p> <p>CC0 1.0 universel (CC0 1.0)</p> <p>http://creativecommons.org/publicdomain/zero/1.0/deed.fr</p> <table border="1"><tr><td><p>La personne qui a associé une œuvre à cet acte a transféré l'œuvre au domaine public en renonçant dans le monde entier à tous ses droits sur l'œuvre selon les lois sur le droit d'auteur, y compris les droits affiliés et connexes, aussi loin que le permet la loi.</p><p>Vous pouvez copier, modifier, distribuer et jouer l'œuvre, même à des fins commerciales, sans avoir besoin d'une permission.</p></td></tr></table>	<p>La personne qui a associé une œuvre à cet acte a transféré l'œuvre au domaine public en renonçant dans le monde entier à tous ses droits sur l'œuvre selon les lois sur le droit d'auteur, y compris les droits affiliés et connexes, aussi loin que le permet la loi.</p> <p>Vous pouvez copier, modifier, distribuer et jouer l'œuvre, même à des fins commerciales, sans avoir besoin d'une permission.</p>
<p>La personne qui a associé une œuvre à cet acte a transféré l'œuvre au domaine public en renonçant dans le monde entier à tous ses droits sur l'œuvre selon les lois sur le droit d'auteur, y compris les droits affiliés et connexes, aussi loin que le permet la loi.</p> <p>Vous pouvez copier, modifier, distribuer et jouer l'œuvre, même à des fins commerciales, sans avoir besoin d'une permission.</p>	

Les serveurs Web peuvent être responsable d'un accès au contenu via le protocole http. Les serveurs Web peuvent représenter un risque significatif pour les raisons suivantes :

- Le port HTTP est souvent visé par des codes malicieux
- Les logiciels de serveur web sont souvent particulièrement complexe avec un long historiques de vulnérabilités.
- Le protocole HTTP n'est pas chiffré et est donc vulnérable à du monitoring passif.

Restrictions des fuites d'informations

Les directives *ServerTokens* et *ServerSignature* déterminent quelles informations le serveur va révéler à propos de la configuration du système.

- *ServerTokens* empêche les informations dans les en-têtes en retournant seulement le mot « Apache ».
- *ServerSignature* empêche de diffuser la version du serveur sur les pages en cas d'erreur. C'est une bonne méthode pour limiter les informations données aux clients.

Ajouter ou corriger les directives suivantes dans */etc/httpd/conf/httpd.conf* afin que le moins d'informations possible soit diffusées.

```
ServerTokens Prod
ServerSignature Off
```

Minimiser les Modules chargeable

Pour plus de renseignements, référez-vous à <http://httpd.apache.org/docs>

Une installation par défaut d'Apache inclut une grande quantité de "Dynamically Shared Objects" (DSO) qui sont chargés lors du lancement. À la différence des modules "*compiled-in*" (ndt: déjà inclus), un DSO peut être désactivé dans le fichier de configuration en supprimant la directive *LoadModule* associée.

Note : un DSO fournit une fonctionnalité supplémentaire seulement si les directives associées sont incluses dans le fichier de configuration Apache. Il faut également noter qu'enlever un DSO produira des erreurs au lancement d'Apache si la configuration contient des directives qui s'appliquent à ce module. Après chaque suppression de DSO, la configuration peut être testé avec la commande suivante pour vérifier que tout fonctionne encore :

```
# service httpd configtest
```

L'utilité de chaque module chargé par défaut sera vérifié un par un. Si aucune des directives du module ne sont utilisées, enlevez-le.

Modules de base d'Apache (Core Modules)

Ces modules comprennent un ensemble basique de modules qui sont probablement requis pour les fonctionnalités d'Apache ; assurez-vous qu'ils ne sont pas commentés dans le fichier `/etc/httpd/conf/httpd.conf`

```
LoadModule auth_basic_module modules/mod_auth_basic.so
LoadModule authn_default_module
modules/mod_authn_default.so
LoadModule authz_host_module modules/mod_authz_host.so
LoadModule authz_user_module modules/mod_authz_user.so
LoadModule authz_groupfile_module
modules/mod_authz_groupfile.so
LoadModule authz_default_module
modules/mod_authz_default.so
LoadModule log_config_module modules/mod_log_config.so
LoadModule logio_module modules/mod_logio.so
LoadModule setenvif_module modules/mod_setenvif.so
LoadModule mime_module modules/mod_mime.so
LoadModule autoindex_module modules/mod_autoindex.so
LoadModule negotiation_module modules/mod_negotiation.so
LoadModule dir_module modules/mod_dir.so
LoadModule alias_module modules/mod_alias.so
```

Si les fonctionnalités présentées ci-dessous ne vous sont pas nécessaire, il est préférable de les commenter dans le fichier de configuration d'Apache afin de les désactiver :

Les différentes lignes se situant dans les cases sont les lignes à commenter selon vos besoins (avec le signe « # » en début de ligne) dans le fichier suivant :

`/etc/httpd/conf/httpd.conf`

Authentification basique HTTP

Les modules suivants sont indispensables si le serveur va fournir du contenu protégé par un mot de passe. L'authentification peut être effectuée en utilisant des fichiers textes (*authn*), des fichiers de mots de passe DBM (*authn dbm*) ou encore un annuaire LDAP.

Le seul module requis par le serveur web dépend de votre choix d'authentification.

```
LoadModule authn_file_module modules/mod_authn_file.so
LoadModule authn_dbm_module modules/mod_authn_dbm.so
```

Le module *authn alias* permet l'authentification basée sur des alias. Le module *authn anon* permet une authentifications anonyme semblable à celle des ftp anonymes. Le module *authz owner* permet d'avoir des autorisations selon les propriétaires du fichier. Le module *authz dbm* permet de gérer les autorisations en se basant sur l'appartenance à des groupes en utilisant l'authentification DBM.

```
# LoadModule authn_alias_module modules/mod_authn_alias.so
# LoadModule authn_anon_module modules/mod_authn_anon.so
# LoadModule authz_owner_module
modules/mod_authz_owner.so
# LoadModule authz_dbm_module modules/mod_authz_dbm.so
```

Authentification HTTP Digest

Ce module fournit des sessions authentifiées et chiffrées. Cependant, ce module est rarement utilisé et considéré comme expérimental. Des méthodes alternatives d'authentification chiffrée sont recommandées, comme SSL.

```
# LoadModule auth_digest_module modules/mod_auth_digest.so
```

mod rewrite

Le module *mod rewrite* est très puissant et peut fournir une protection contre certaines attaques web. Cependant, il est également très complexe et a un passé relativement lourd de vulnérabilités lui-même.

```
# LoadModule rewrite_module modules/mod_rewrite.so
```

Support LDAP

Ce module fournit une authentification HTTP via un annuaire LDAP.

```
# LoadModule ldap_module modules/mod_ldap.so
#LoadModule authnz_ldap_module modules/mod_authnz_ldap.so
```

Si vous utilisez LDAP, il est également préférable d'utiliser le chiffrement SSL.

Server Side Includes

Les *Server Side Includes* permettent de générer des pages web dynamiquement via l'insertion de code côté serveur. Cependant, cette technologie est obsolète et crée de sérieux problèmes de sécurité.

```
# LoadModule include_module modules/mod_include.so
```

Si vous avez vraiment besoin des *Server Side Includes*, vous devez les activer avec l'option « *IncludesNoExec* » pour éviter l'exécution arbitraire de code. Les données entrées par l'utilisateur doivent aussi être encodées afin d'éviter les failles de sécurité inter-sites (XSS).

MIME Magic

Ce module ajoute une couche de support MIME qui est certainement en trop dans la plupart des configurations.

```
# LoadModule mime_magic_module modules/mod_mime_magic.so
```

WebDAV (Identification et Versionnement distribués)

WebDAV est une extension du protocole HTTP qui donne un accès "distribué et "collaboratif" au Web.

Suite aux nombreux problèmes de sécurité concernant WebDAV, son utilisation n'est pas recommandée.

```
# LoadModule dav_module modules/mod_dav.so
# LoadModule dav_fs_module modules/mod_dav_fs.so
```

Si il y a un besoin critique pour WebDAV, énormément d'attention doit être portée à sa configuration. Vu que l'accès DAV permet aux clients à distance de manipuler les fichiers du serveur, toutes les zones du serveur où DAV est activé doit être protégée par des authentifications chiffrées.

Statut de l'activité du serveur (Server Activity Status)

Ce module fournit un accès en temps réel à des statistiques d'opérations internes au serveur web. Cette fuite d'informations n'est pas nécessaire et doit être désactivée.

```
# LoadModule status_module modules/mod_status.so
```

Si il y a un besoin critique pour ce module, assurez-vous que l'accès à la page

est uniquement accessible à un ensemble d'hôtes dans la configuration du gestionnaire d'état.

Web Server Configuration Display

Affichage de la configuration du serveur web

Ce module crée une page web pour illustrer la configuration du serveur. C'est une faille de sécurité possible et il doit être désactivé.

```
# LoadModule info_module modules/mod_info.so
```

Si il y a un besoin critique de ce module, utilisez la directive *Location* pour fournir une liste de contrôle d'accès pour restreindre les accès aux informations.

Correction d'URLs sur les fautes de frappes

Ce module tente de trouver un document en autorisant une faute de frappe dans une requête échouée.

```
# LoadModule spelling_module modules/mod_spelling.so
```

Cette fonctionnalité affaiblit la sécurité du serveur en permettant le parcours du site plus facilement.

Dossiers utilisateurs

La directive *UserDir* fournit des dossiers par-utilisateur, en permettant des URLs basées sur des noms d'utilisateurs.

```
# LoadModule userdir_module modules/mod_userdir.so
```

S'il vous avez absolument besoin de ce module, incluez la ligne « *UserDir disabled root* » (au moins) dans le fichier de configuration. Dans l'idéal, il est préférable de désactiver *UserDir*, puis de l'activer au cas par cas pour des utilisateurs précis qui en ont besoin.

Note : les utilisateurs d'un serveur web peuvent être facilement énumérés en utilisant ce module.

Support des proxy

Ce module fournit un support des proxys, en autorisant Apache à rediriger des requêtes et servir de portail vers d'autres serveurs.

```
# LoadModule proxy_module modules/mod_proxy.so
```

```
# LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
# LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
# LoadModule proxy_http_module modules/mod_proxy_http.so
# LoadModule proxy_connect_module modules/mod_proxy_connect.so
```

Si le support des proxies est requis, chargez le module proxy et le gestionnaire associé au protocole approprié (un parmi *proxy http*, *proxy ftp*, ou *proxy connect*). En sus, assurez-vous qu'un serveur est sécurisé avant d'activer la proxyfication, étant donné que les serveurs de proxy sont un risque de sécurité potentiel. Étant donné que *mod status* n'est pas recommandé, le module *proxy balancer* ne devrait également pas être utilisé.

Support du cache

Ce module permet à apache de mettre des données en cache, optimisant ainsi l'accès au contenu fréquemment demandé. Cependant, non seulement ce module est expérimental, mais il crée également des failles de sécurité potentielles, comme la possibilité d'outrepasser les directives *Allow* et *Deny*.

```
# LoadModule cache_module modules/mod_cache.so
# LoadModule disk_cache_module modules/mod_disk_cache.so
# LoadModule file_cache_module modules/mod_file_cache.so
# LoadModule mem_cache_module modules/mod_mem_cache.so
```

Si le cache est requis, il ne doit pas être activé sur le contenu pour lequel l'accès est restreint.

Support des CGI (et modules associés)

Ce module permet au HTML d'interagir avec le langage de programmation web CGI.

```
# LoadModule cgi_module modules/mod_cgi.so
# LoadModule env_module modules/mod_env.so
# LoadModule actions_module modules/mod_actions.so
# LoadModule suexec_module modules/mod_suexec.so
```

Si le serveur web a besoin des CGI, activez le module *cgi*. Si des fonctionnalités CGI étendues sont requises, incluez les modules appropriés. Le module *env* permet de contrôler l'environnement fourni aux scripts CGI. Le module *Actions* permet aux événements CGI d'être appelés quand des fichiers d'un certain type sont demandés. Le module *su exec* permet de lancer les scripts CGI en tant qu'un utilisateur ou groupe particulier, au lieu de l'utilisateur ou groupe du serveur web.

Composants optionnels divers

Les modules suivants accomplissent des tâches très spécifiques, parfois en donnant accès à juste quelques directives supplémentaires. Si cette fonctionnalité n'est pas requise (ou si vous n'utilisez pas ces directives), commentez le module associé :

Filtrage externe (la réponse passe à travers un programme externe avant d'être envoyée au client)

```
# LoadModule ext_filter_module modules/mod_ext_filter.so
```

- Gestion du cache par l'utilisateur

```
# LoadModule expires_module modules/mod_expires.so
```

- Compression des données sortantes (permet de compresser le contenu avant l'envoi au client)

```
# LoadModule deflate_module modules/mod_deflate.so
```

- Personnalisation des en-têtes HTTP

```
# LoadModule headers_module modules/mod_headers.so
```

- Gestion de l'activité des utilisateurs via des cookies

```
# LoadModule usertrack_module modules/mod_usertrack.so
```

- Configuration dynamique de serveurs virtuels

```
# LoadModule vhost_alias_module modules/mod_vhost_alias.so
```

Limiter l'inclusion des fichiers de configuration

La directive *Include* indique à Apache de charger des fichiers de configuration supplémentaires depuis un chemin donné. Le fichier de configuration par défaut charge tous les fichiers qui finissent par `.conf` depuis le dossier `/etc/httpd/conf.d`

Pour éviter la surcharge de configurations, la ligne suivante doit être commentée et remplacée avec des directives *Include* qui ne référencent que des fichiers de configuration requis.

```
# Include conf.d/*.conf
```

Si la modification ci-dessus a été faite, assurez vous que le chiffrement SSL reste chargé en incluant explicitement le fichier de configuration correspondant :

```
Include conf.d/ssl.conf
```

Si PHP est requis, une modification similaire doit être effectuée :

```
Include conf.d/php.conf
```

Restrictions de dossiers

Le tag *Directory* dans le fichier de configuration du serveur web permet un contrôle d'accès plus précis pour un dossier précis. Tous les dossiers web doivent être configurés au cas par cas, en n'autorisant l'accès que si nécessaire.

Restreindre l'accès au dossier racine (root)

Le dossier root Apache devrait toujours avoir la configuration la plus "restrictive" activé.

```
<Directory / >  
Options None  
AllowOverride None  
Order allow,deny  
</Directory>
```

Utiliser les modules appropriés pour améliorer la sécurité d'Apache

Parmi les modules disponibles pour Apache, il en existe plusieurs qui peuvent améliorer la sécurité de l'installation du serveur Web. Cette section recommande et aborde le déploiement des modules liés à la sécurité.

Déployer mod ssl

Parce que le trafic HTTP est un protocole en texte brut, tout ce qui y transite est susceptible d'être surveillé de manière passive. Si jamais un besoin de confidentialité est avéré, SSL devrait être configuré pour vous permettre de chiffrer le contenu.

Note : le module *mod nss* est une alternative certifiée FIPS 140-2 du mod *ssl*. Ces modules partagent une grande partie de leur code source et sont quasiment identique en termes de fonctionnalités. Si une validation FIPS 140-2 est demandée, *mod nss* devrait être utilisé. Si certaines de ses fonctionnalités ou une meilleure compatibilité sont nécessaires, *mod ssl* est à préférer.

- Installer mod_ssl :

```
yum install mod_ssl
```

- Créez un certificat SSL

Sur votre CA (si vous utilisez le vôtre) ou un autre système sécurisé, générez une paire de clés pour votre serveur web:

```
cd /etc/pki/tls/certs  
openssl genrsa -des3 -out httpserverkey.pem 2048
```

Lorsque demandé, saisissez une phrase secrète complexe pour protéger la paire de clefs du serveur web.

- Générez une demande de signature de certificat (*Certificate Signing Request* ou CSR) à partir de la clef pour la CA.

```
openssl req -new -key httpserverkey.pem -out httpserver.csr
```

Saisissez la phrase secrète pour la paire de clés du serveur web et remplissez les champs le mieux possible (ou appuyez sur Entrée pour accepter les valeurs par défaut); le nom commun (Common Name) est particulièrement important. Il doit correspondre exactement au nom de domaine de votre serveur (par exemple www.exemple.com) pour que le certificat fonctionne.

Le fichier /etc/pki/openssl.conf détermine quels autres champs (par exemple le Pays, nom d'Organisation, nom, etc) doivent correspondre entre la requête du serveur et la CA. Laissez le mot de passe de "challenge" FIXME et le nom de société optionnel blancs. Ensuite, la CSR du serveur Web doit être signée pour créer le certificat du serveur web. Vous pouvez soit envoyer la CSR à une CA reconnue ou la signer avec votre CA.

- Signer httpserver.csr avec votre CA :

```
openssl ca -in httpserver.csr -out httpservercert.pem
```

Lorsqu'on vous le demande, saisissez la phrase secrète pour continuer et finir le traitement. Le certificat httpservercert.pem nécessaire à l'activation de SSL sur le serveur web est alors dans le repertoire courant.

Enfin, la clef du serveur web et le fichier du certificat doivent être déplacés sur le serveur web. Utilisez des médias amovibles si possible. Mettez la clef sur serveur et le fichier de certificat dans /etc/pki/tls/http en les nommant serverkey.pem et servercert.pem.

Installer le certificat SSL

Ajoutez ou modifiez le fichier de configuration `/etc/httpd/conf.d/ssl.conf` pour avoir ceci :

```
# establish new listening port
Listen 443
# seed appropriately
SSLRandomSeed startup file:/dev/urandom 1024
SSLRandomSeed connect file:/dev/urandom 1024
<VirtualHost site-on-certificate.com:443>
# Enable SSL
SSLEngine On
# Path to server certificate + private key
SSLCertificateFile /etc/pki/tls/http/servercert.pem
SSLCertificateKeyFile /etc/pki/tls/http/serverkey.pem
SSLProtocol All -SSLv2
# Weak ciphers and null authentication should be denied unless
absolutely necessary (and even then, such cipher weakening should occur
within a Location enclosure)
SSLCipherSuite HIGH:MEDIUM:!aNULL:+MD5
</VirtualHost>
```

Assurez vous que tous les dossiers qui contiennent du contenu SSL sont accessibles uniquement via SSL dans le fichier `/etc/httpd/conf/`

```
<Directory /var/www/html/secure>
# require SSL for access
SSLRequireSSL
SSLOptions +StrictRequire
# require domain to match certificate domain
SSLRequire %{HTTP_HOST} eq "site-on-certificate.com"
# rather than reply with 403 error, redirect user to appropriate site
# this is OPTIONAL - uncomment to apply
# ErrorDocument 403 https://site-on-certificate.com
</Directory>
```

Mettre en place le Mod Security

Mod security fournit une application Firewall pour Apache. Suivez l'installation de *mod security* avec les règles de base, des configurations spécifiques peuvent être trouvées sur <http://www.modsecurity.org> pour élaborer des règles qui conviennent mieux à la sécurité que demandent des applications Web.

- [Installer le mod security](#) (pour Red Hat)

```
yum install mod_security
```

- Configurer le filtrage de mod security

Mod_security supporte un nombre important d'options, bien trop pour les couvrir entièrement dans ce guide. Cependant, la liste suivante contient un ensemble plus restreint de filtres que nous suggérons d'ajouter au fichier `/etc/httpd/conf/httpd.conf` :

```
# enable mod security
SecFilterEngine On
# enable POST filtering
SecFilterScanPost On
# Make sure that URL encoding is valid
SecFilterCheckURLEncoding On
# Accept almost all byte values
SecFilterForceByteRange 1 255
# Prevent directory traversal
SecFilter "\.\/"
# Filter on specific system specific paths
SecFilter /etc/passwd
SecFilter /bin/
# Prevent cross-site scripting
SecFilter "<[[:space:]]* script"
# Prevent SQL injection
SecFilter "delete[[:space:]]+from"
SecFilter "insert[[:space:]]+into"
SecFilter "select.+from"
```

Utiliser les modules de protection contre les Défis de service (DOS)

Les attaques via déni de service sont difficiles à détecter et éviter tout en maintenant un accès convenable pour les utilisateurs légitimes. Cependant, il y a un certain nombre de modules de gestion de trafic qui tentent de résoudre ce problème. Les modules populaires de protection contre le DoS sont, entre autres :

```
mod_throttle mod_bwshare mod_limitipconn mod_dosevasive
```

Il est recommandé d'implémenter la protection contre le déni de service pour le serveur web. Cependant, ce guide laisse les détails de configuration spécifique à charge du lecteur.

Configurer correctement les modules supplémentaires

Toute fonctionnalité ajoutée au serveur web via des modules

supplémentaires doit être configurés correctement.

Configurer PHP de manière sécurisée

PHP est un langage de script côté serveur très utilisé, et souvent mal configuré. Il devrait être utilisé avec précaution, mais configuré correctement si nécessaire.

Effectuez les modifications suivantes sur le fichier `/etc/php.ini`

```
# Do not expose PHP error messages to external users
display_errors = Off
# Enable safe mode
safe_mode = On
# Only allow access to executables in isolated directory
safe_mode_exec_dir = php-required-executables-path
# Limit external access to PHP environment
safe_mode_allowed_env_vars = PHP_
# Restrict PHP information leakage
expose_php = Off
# Log all errors
log_errors = On
# Do not register globals for input data
register_globals = Off
# Minimize allowable PHP post size
post_max_size = 1K
# Ensure PHP redirects appropriately
cgi.force_redirect = 0
# Disallow uploading unless necessary
file_uploads = Off
# Disallow treatment of file requests as fopen calls
allow_url_fopen = Off
# Enable SQL safe mode
sql.safe_mode = On
```

Configurer le système d'exploitation pour protéger le serveur web

Les étapes de configuration qui suivent doivent être effectuées sur la machine qui héberge le serveur web, afin qu'il soit placé dans l'environnement le plus sécurisé possible.

Restreindre l'accès aux fichiers et aux dossiers

Limiter l'accès aux fichiers et dossier critiques d'Apache :

```
chmod 511 /usr/sbin/httpd
chmod 750 /var/log/httpd/
chmod 750 /etc/httpd/conf/
chmod 640 /etc/httpd/conf/*
chgrp -R apache /etc/httpd/conf
```

Configurer iptables pour permettre l'accès au serveur web

Éditez `/etc/sysconfig/iptables`. Ajoutez les lignes suivantes, en vous assurant qu'elles apparaissent avant les dernières lignes LOG et DROP pour la chaîne RH-Firewall-1-INPUT :

```
-A RH-Firewall-1-INPUT -m state --state NEW -p tcp --dport 80 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -p tcp --dport 443 -j ACCEPT
```

La configuration iptables par défaut ne permet pas de connexion entrante sur les ports HTTP (80) et HTTPS (443) utilisés par le serveur web. Cette modification permet l'accès, tout en laissant les autres ports du serveur dans leur état protégé par défaut.

Lancer Apache dans un chroot si possible

Mettre Apache dans un chroot réduit les dégâts causés par une intrusion potentielle dans le système en isolant le serveur web dans une petite section du système de fichiers.

Afin de configurer Apache pour qu'il se lance depuis un chroot, éditez le fichier de configuration `/etc/httpd/conf/httpd.conf`, et ajoutez la ligne :
`SecChrootDir /chroot/apache`

Il est également indispensable de placer tous les fichiers requis par apache dans le système de fichiers chrooté sur `/chroot/apache`, ce qui inclut les binaires d'Apache, les modules, fichiers de configuration, et pages web. Les détails de cette configuration vont bien au-delà de la portée de ce guide.