


10 oct 2008

# Sécurité avancée du serveur web Apache : mod_security et mod_dosevasive

Catégorie : Web Tags : misc

~~Retrouvez cet article dans : **Misc 20**~~

Le serveur Apache n'est plus à présenter et propose nombre d'options liées à la sécurité en soi [1]. Pour autant, des modules tiers permettent d'étendre ses possibilités, que ce soit en termes de filtrage des requêtes et des réponses ou pour contrer les attaques visant à surcharger le serveur. Cet article présente deux de ces modules : mod_security et mod_dosevasive. Tout au long de l'article, nous prendrons l'exemple d'un proxy entrant, mais ces exemples sont bien sûr facilement transposables à un serveur web.

## mod_security : validation et filtrage des échanges

mod_security [2] permet de contrôler de manière très fine les échanges entre le client et le serveur : il filtre ces échanges et autorise également une journalisation complète des événements. Nous parlons principalement dans cette section de la version stable (1.8.X) de ~~mod_security~~.
Les approches « habituelles » en termes de filtrage et de validation des échanges HTTP se concentrent souvent uniquement sur les requêtes. En particulier, Apache permet déjà de filtrer les requêtes selon la méthode HTTP utilisée [3]. ~~mod_security~~ va beaucoup plus loin en travaillant également sur les réponses HTTP et en permettant une analyse beaucoup plus étendue des requêtes (et notamment du contenu complet des requêtes de type POST).
S'intégrant avant le traitement normal des requêtes par Apache (en particulier pour Apache 2.0), ~~mod_security~~ est également capable de filtrer les flux HTTPS et de corriger les URL mal formés.
Enfin, last but not least, ce module est un Logiciel Libre publié sous licence GPL et écrit par Ivan Ristic [4].

### Validation des requêtes

Avant de procéder au filtrage en soi des URL reçus, ~~mod_security~~ applique quelques transformations à ces dernières. Ces transformations permettent de s'affranchir d'un certain nombre d'astuces utilisées par les attaquants qui réussissent ainsi à passer outre certains filtres. Prenons par exemple le cas où l'on souhaite bloquer toutes les requêtes contenant ~~/bin/sh~~. Un attaquant un peu malin tentera des approches biaisées, du style ~~/bin/./sh~~... Un filtre digne de ce nom se doit donc de valider les réductions

possibles d'URL et d'interpréter au minimum ces réductions. C'est exactement la portée des transformations dont nous parlons ici.
Les transformations suivantes sont appliquées :

- Si le filtre tourne sous Windows, transformation des caractères \ en /.
- Transformation de /./ en /.
- Optionnellement, validation de l'encodage de l'URL.
- Optionnellement, restriction de la plage de caractères ASCII (table étendue) autorisée. mod_security valide ces caractères selon leur valeur, comprise en décimal entre 0 et 255.
- Transformation de // en /.

### Rappel sur http

Toute requête HTTP est basée sur le format d'échange de données MIME (Multipurpose Internet Mail Extension), qui comporte un en-tête et un corps (comme pour les emails en SMTP). Dans le cas d'une requête HTTP, seul l'en-tête est utilisé, le corps reste vide (sauf pour les requêtes de type POST). Les réponses HTTP quant à elles utilisent les deux. Dans tous les cas, en-tête et corps sont séparés par une ligne vide. Voici un exemple de requête http :

```
GET /index.html HTTP/1.1
Host: www.miscmag.com
User-Agent: foobar
Referer: www.kernel.com
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
```

Depuis la version 1.1 de HTTP, les deux premières lignes de notre en-tête HTTP d'exemple sont impératives. Les autres lignes sont indicatives ou servent à la négociation de contenu (préférences en formats multimédias, en termes de langage et autres). Voici une réponse possible du serveur :

```
HTTP/1.1 200 OK
Date: Thu, 05 May 2005 10:13:21 GMT
Server: Apache/2.0.54 (Debian GNU/Linux) PHP/4.3.10-12
Content-Length: 305
Content-Type: text/html; charset=iso-8859-1
<html>Bonjour</html>
```

Le serveur confirme qu'il parle HTTP en version 1.1, nous donne un code de succès (200) et spécifie diverses informations, dont la plus importante est le champ Content-Type. Le navigateur utilise en effet ce champ pour interpréter la réponse. Ici, la valeur text/html de ce champ indique au navigateur d'interpréter cette page comme du code HTML.

D'autres transformations sont également effectuées. Notamment, le caractère NULL (\0) est transformé en espace.
À partir de ce point, les filtres que nous définissons peuvent être appliqués avec plus de confiance. Lançons-nous dans la définition d'un hôte virtuel, proxifié en HTTPS et filtré par mod_security :

```
<VirtualHost *:443>
 ServerName www.foo.com

 #Mod_security est activé pour ce VirtualHost
 SecFilterEngine On

 #Nous filtrons également le contenu complet des requêtes POST
 SecFilterScanPost On
  #Seuls les corps de requêtes de l'un de ces deux types sont acceptés (ceci
       #concerne uniquement les requêtes POST, les autres requêtes n'ont pas de corps
       SecFilterSelective HTTP_Content_Type "!(^$|application/x-www-form-                    urlencoded$|^multipart/form-data;)"

       #Bloquage de l'encodage par morceau des requêtes - utilisé uniquement pour des
       #attaques - non implémenté par les navigateurs
       SecFilterSelective HTTP_Transfer-Encoding "!^$"

       #Vérifions que seuls des encodages valides sont reçus
       SecFilterCheckURLEncoding On
```

```
        #Vérifions que l'encodage est conforme à Unicode - à utiliser seulement si
        #notre application web utilise Unicode
        SecFilterCheckUnicodeEncoding On

        #Que faire par défaut quand nos filtres correspondent à une requête. Nous
        #journalisons l'événement et renvoyons une erreur 404
        SecFilterDefaultAction "deny,log,status:404"

        #Vérifions le format des Cookies reçus
        SecFilterCheckCookieFormat On
        #Tâchons de corriger les cookies invalides que nous recevons - attention, ceci
        #peut "casser" certaines applications et est donc à tester avec précautions
        SecFilterNormalizeCookies On

        #Les deux directives Apache qui font le "vrai" travail : proxyfier les requêtes
        ProxyPass / http://192.168.1.0/ #Adresse IP du serveur protégé
        ProxyPassReverse / http://192.168.1.0/
</VirtualHost>
```

Cet exemple présente une configuration « de base » du module ~~mod_security~~ ; le gros du travail : écrire les règles de filtrage, reste à faire. Il s'agit bien entendu de la partie la plus délicate, puisque la pertinence des informations à positionner dans cette partie dépend (parfois) de notre système d'exploitation, de l'application utilisée, etc.

### Quelques schémas d'attaque HTTP

De nombreuses attaques sont possibles au moyen du protocole HTTP et sont bien sûr plus ou moins efficaces selon la configuration et la version du serveur :

*Accès au shell*

Un grand classique est de tenter d'ouvrir un shell sur la machine afin de pouvoir lui passer des commandes au choix de l'attaquant.
Pour Windows :

```
GET /scripts/..%255c%255c../winnt/system32/cmd.exe?/
c+dir HTTP/1.1
```

Pour Unix :

```
GET /../../../../bin/sh HTTP/1.1
```

*Injection SQL [5]*

Ce type d'attaque suppose que le serveur web s'appuie sur une base de données et vise à envoyer des données qui seront passées à la base de données pour en extraire des données ou passer outre un test :

```
GET /identify.php?username=joe&pass=bah'%20OR%20'x'='x HTTP/1.1
```

Dans ce cas trivial (plutôt codé en requête POST en « vrai »), le script PHP « naïf » récupère le nom d'utilisateur et le mot de passe en variables, puis réalise une requête SQL sur la base de données, ce qui dans notre cas donnera par exemple :

```
SELECT * from users where username='joe' and password='bah' OR 'x'='x';
```

*Cross site scripting [6]*

Plus récente et très différente, cette attaque utilise simplement le fait que le serveur accepte des contenus HTML ou Javascript et les réaffiche. On pourra ainsi injecter du code Javascript malicieux sur un serveur afin de compromettre des utilisateurs légitimes du serveur. Ce type d'attaque ne vise ni ne compromet directement le serveur.

## Règles de filtrage des requêtes

Bloquons toute tentative d'accès au shell de la machine en ajoutant dans la définition du virtualhost :

```
SecFilter /bin/sh
```

Il s'agit de la méthode la plus élémentaire de filtrage, qui fonctionne par mot clé. Le contenu spécifié à la suite de cette directive est comparé à la première ligne de la requête HTTP. Si la requête est une requête de type POST et que nous avons positionné l'option SecFilterScanPost On comme ci-avant, chaque ligne du corps de la requête fait également l'objet de cette comparaison. SecFilter peut également fonctionner de manière plus étendue et accepte comme argument une expression rationnelle :

```
SecFilter "!(php|asp)"
```

bloque toute requête ne contenant pas le mot ~~php~~ ou ~~asp~~.
~~SecFilter~~ constitue l'introduction idéale aux capacités de filtrage de ~~mod_security~~, mais l'on s'aperçoit assez rapidement que son utilisation est souvent trop large. Supposons par exemple que nous gérons un forum dédié aux programmeurs du Shell... Il sera courant de voir quelqu'un envoyer une requête de type POST contenant notre chaîne ~~/bin/sh~~... qui en termes de sécurité, et en première approche, est à prohiber uniquement dans la première ligne de la requête. La directive ~~SecFilterSelective~~ prend un paramètre de plus, qui spécifie la « zone » où notre chaîne interdite est prohibée. ~~mod_security~~ met à disposition un grand nombre de telles « zones », qu'il convient de présenter, à défaut de pouvoir en donner ici la liste exhaustive [7] :

- Contenu de la requête, envoyé par le client : par exemple ~~REMOTE_USER~~, ~~POST_PAYLOAD~~ ou ~~REQUEST_URI~~
- Paramètres liés à la requête, indépendants du client : ~~SERVER_SOFTWARE~~, ~~DOCUMENT_ROOT~~
- Paramètres plus ou moins extérieurs à la requête : ~~TIME_DAY~~, ~~API_VERSION~~
- Paramètres ajustables à souhait : ~~HTTP_nom_de_header~~, ~~ENV_nom_de_variable~~

Les utilisateurs avancés d'Apache reconnaîtront parmi les exemples un grand nombre de noms de variables internes à Apache, utilisables notamment par ~~mod_rewrite~~. Nous retrouvons ici la souplesse de cet outil, dans un environnement plus spécifique puisque dédié à la sécurité.
Pour revenir à notre exemple de forum, nous désirons filtrer les requêtes tentant un accès au shell, et ce uniquement dans la première ligne de la requête. Pour cet exemple, nous pouvons donc positionner :

```
SecFilterSelective REQUEST_URI /bin/sh
```

## Filtrage des réponses HTTP

Filtrer les requêtes est une bonne chose, mais nous pouvons faire plus : appliquer certains critères de recherche aux réponses renvoyées par le serveur web. Il suffit pour cela d'utiliser la directive ~~SecFilterSelective~~, avec comme « endroit » ~~OUTPUT~~.

```
#Nous filtrons les réponses du serveur HTTP au client
SecFilterScanOutput On
#Filtrons uniquement des mots clés sur les documents dont le type MIME est pertinent
#Par exemple, filtrer des mots clés dans une image ou un fichier audio est sans objet
SecFilterOutputMimeTypes "(null) text/plain text/html"
#Ne laissons pas sortir "information(s) confidentiel(les)"
SecFilterSelective OUTPUT "informations* confidentiel*e*s*"
#Ne laissons pas le client constater que PHP a renvoyé une erreur fatale, si cela arrive
SecFilterSelective OUTPUT "Fatal error:" deny,status:404
```

Comme montré dans le tout dernier exemple, les directives ~~SecFilter~~ ainsi que ~~SecFilterSelective~~ acceptent un argument supplémentaire, qui correspond aux actions à réaliser en cas de correspondance entre la règle et le contenu HTTP. Si aucune action n'est spécifiée, les actions définies par la directive ~~SecFilterDefaultAction~~ sont appliquées.
Bien entendu, les règles de filtrage appliquées aux réponses sont utilisées à un autre moment que les règles vues auparavant : il faut que le serveur ait généré une réponse, et soit prêt à l'envoyer, pour que nous filtrions celle-ci.
Attention toutefois : le filtrage des réponses demande que ~~mod_security~~ garde en mémoire la page de réponse entière avant de l'envoyer au client, ce qui peut consommer de grandes quantités de mémoire, en particulier sur un serveur chargé et si les pages servies sont d'un volume important. Il convient donc

d'évaluer l'impact en termes de performances de l'utilisation de ces possibilités. Pour l'instant aucune directive de mod_security ne permet de limiter la taille des réponses ; ce manque devrait être pallié dans une prochaine version.

## Les actions utilisables par le filtre

Les actions suivantes peuvent être utilisées dans les directives SecFilter, SecFilterSelective, aussi naturellement que SecFilterDefaultAction :

- pass : le contenu poursuit son chemin parmi les filtres à suivre ;
- allow : le contenu est accepté, sans passer par les autres filtres à suivre ;
- deny : interruption du traitement de la requête, envoi d'un code d'erreur au client ;
- status : suivi d'un code d'erreur HTTP, renvoie le code d'erreur spécifié au client ;
- redirect : redirige le client vers l'URL spécifié ;
- exec : lance le binaire spécifié, en lui passant toutes les variables d'environnement courantes ;
- log : journalise la correspondance de la règle dans le fichier d'erreur d'Apache ;
- nolog : l'inverse de ci-avant ;
- skipnext : les prochaines règles de filtrage (nombre à spécifier en argument) sont ignorées pour cette requête ;
- chain : la requête doit correspondre à la présente règle, mais aussi à la suivante, pour que la décision de cette dernière soit appliquée ;
- pause : attend un certain nombre de millisecondes (spécifié en paramètre) avant de répondre à cette requête. Attention aux performances en utilisant ce paramètre !

Il est courant de spécifier comme action par défaut quelque chose du genre :

```
SecFilterDefaultAction "deny,log,status:404"
```

comme nous l'avons positionné en début de configuration. Par ailleurs, les actions skipnest et surtout chain sont particulièrement intéressantes pour conjuguer des règles entre elles.
L'action pass peut paraître inutile à première vue ; elle est en réalité quasiment toujours utilisée en conjonction avec l'action log pour journaliser la correspondance d'une requête sans la bloquer, par exemple :

```
SecFilter "credit card" "pass,log"
```

## Fonctionnalités chroot de mod_security

Il est bien sûr possible de chrooter Apache sans utiliser mod_security, mais l'utilisation de ce module nous simplifie considérablement le travail. En effet, positionnez la directive :

```
SecChrootDir /repertoire/a/utiliser
```

et le serveur web sera chrooté. Ceci permet d'utiliser chroot, sans copier les binaires et bibliothèques d'Apache dans le chroot ! Seul le contenu web doit y être positionné, le cas échéant. Dans le cas d'exemple couvert par cet article, nous configurons un proxy entrant, aussi aucun contenu n'est à prévoir.
Pour Apache 1.3, l'ordre des modules est très important pour l'utilisation de cette directive : mod_security doit être chargé avant les autres modules. Sous Apache 2.0, l'ordre de chargement est géré de manière interne, si bien que l'ordonnancement des modules est sans importance.
Bien sûr, si cette fonctionnalité est utilisée sur un serveur hébergeant des scripts CGI ou un langage interprété comme PHP, les binaires et bibliothèques utilisés devront être maintenus dans le chroot.

## Autres fonctionnalités de mod_security

Voici, brièvement présentées, quelques possibilités (parmi d'autres) proposées par mod_security. Le lecteur intéressé trouvera des descriptions plus détaillées dans la documentation du projet.

### Traitement des fichiers uploadés

mod_security fait toujours passer les fichiers uploadés par les utilisateurs par un répertoire temporaire, paramétrable au moyen de la directive SecUploadDir. L'administrateur peut demander à mod_security de conserver les fichiers uploadés dans ce répertoire temporaire, au moyen de SecUploadKeepFiles On. La directive SecUploadApproveScript est également très intéressante : elle permet de lancer un exécutable (comme un script lançant lui-même un antivirus) pour valider le fichier uploadé. La documentation de mod_security fournit également un tel script.

## Dialogue avec le pare-feu

Comme vu ci-dessus, l'action exec, passée à la directive SecFilter ou SecFilterSelective, permet de demander l'exécution d'un programme au choix lorsqu'une requête correspond à un filtre. Il est donc bien entendu possible d'utiliser cette fonctionnalité pour ajouter des adresses IP source en liste noire, dont nous refuserons les accès à notre serveur, voire à notre réseau. Attention cependant, certaines précautions sont à considérer pour ce type d'utilisation (comme pour d'autres filtres tel PortSentry) :

- Si l'attaquant passe par un serveur mandataire, c'est alors l'adresse du mandataire qui sera listée, bloquant également tous les autres utilisateurs du mandataire.
- Si l'attaquant réussit à usurper l'adresse d'une de nos passerelles, il pourrait nous isoler complètement d'un réseau voire d'Internet.
- Si l'attaquant dispose d'une adresse IP non fixe, il lui suffira de changer d'adresse pour pouvoir nous attaquer de nouveau et nous bloquerons en plus les éventuels accès d'un nouvel utilisateur légitime.

### Mesures de performances

Dans la version de développement (1.9) du module, trois variables d'environnement sont créées et visibles par Apache : il s'agit de mod_security-time1, mod_security-time2 et mod_security-time3, qui spécifient respectivement les temps suivants (en microsecondes) : fin de l'initialisation de mod_security, fin de l'application des règles définies par l'administrateur, fin de génération de la réponse (par Apache). Cette possibilité reste pour l'instant expérimentale et sera déployée dans une prochaine branche stable de mod_security.
À l'exception des trois variables que nous venons de décrire, l'ensemble de cet article est valide pour la version stable (1.8) du module. Pour l'instant, la version de développement (1.9) apporte de nouvelles variables spécifiables dans les zones de filtrage, permet l'utilisation (déconseillée) de directives de filtrage dans les fichiers .htaccess et modifie quelques paramétrages par défaut.
Toutes vulnérabilités web, du débordement de tampon à l'injection de code SQL, en passant par les attaques de Cross Site Scripting, peuvent sur le papier être protégées par une configuration bien pensée de mod_security, ce qui est effectivement faisable, au prix d'un effort minutieux de repérage des failles. En pratique, il est difficile de réaliser un tel repérage et donc d'être sûr que les applications sont parfaitement sécurisées. L'idéal reste, bien entendu, de valider les applications web utilisées elles-mêmes avant de les déployer sur un serveur web.
Grâce au script snort2modsec.pl [8], qui permet de traduire des règles Snort [9] en règles de filtrage utilisables par mod_security, le module peut assez facilement être préconfiguré pour bloquer un certain nombre d'attaques bien connues.

# mod_dosevasive

mod_dosevasive est un deuxième module de sécurité pour Apache, qui apporte des possibilités très différentes de celles que nous venons d'examiner. Son objectif est en effet de détecter les accès massifs révélant des attaques de déni de service [distribué] ([d]DOS) ou des attaques de force brute, et de prendre des mesures contre ceux-ci. Bien entendu, le but est également que les utilisateurs légitimes puissent continuer d'utiliser le service dans des conditions normales.
Disponible lui aussi sous licence GPL, ce module est écrit par Jonathan A. Zdziarski [10]. Ce module peut être utilisé avec Apache 1.3 ou 2.0.

# Principe de fonctionnement

### Détection des attaques

La détection d'une attaque de type « déni de service » est complexe : chaque requête, prise séparément, pourrait souvent être considérée comme légitime. Il convient donc de distinguer une attaque, par définition de source malveillante, d'un éventuel pic de trafic, qui pourrait résulter d'un référencement sur Slashdot par exemple.
~~mod_dosevasive~~ [11] considère qu'une attaque est en cours et positionne l'adresse IP source en liste noire si l'une de ces conditions est remplie :

- Que la même ressource fait l'objet de requêtes répétées (plusieurs par secondes) depuis la même adresse IP source. Ceci permet d'écarter les consultations régulières puisqu'un utilisateur bienveillant ne récupère normalement pas plusieurs fois par seconde une même ressource.
- Que la même adresse IP réalise plus de N requêtes par seconde sur le même processus enfant d'Apache. Une consultation normale se limite à la récupération de quelques objets à la foi.
- Qu'une adresse déjà en liste noire fait une requête. Cette troisième condition paraît étrange à première vue, mais s'éclaire quand on sait que la liste noire est temporaire : une adresse IP référencée dans cette liste en est retirée après un certain délai configurable si elle ne réalise aucune requête sur le serveur durant ce délai.

La combinaison de ces trois facteurs permet de détecter de manière fiable une attaque en cours.

### Actions entreprises à l'égard des attaquants

Dès lors qu'une adresse IP est positionnée dans la liste noire, toutes ses requêtes feront immédiatement l'objet d'une réponse de type 403, sans que le serveur web ne réalise le traitement normal de la requête. Ceci économise notablement les ressources, en particulier si l'attaquant fait appel à des ressources CGI ou à des contenus dynamiques.

# Pertinence de la méthode

L'expérience a montré que ce module répond en soi aux besoins de manière très efficace pour des dénis de services petits ou moyens, qu'ils aient pour origine une seule ou plusieurs machines. Réagir à de plus importantes attaques est un problème beaucoup plus avancé : si l'attaquant dispose de suffisamment de ressources réseau pour remplir le tuyau d'entrée du réseau attaqué, il devient beaucoup plus difficile de le contrer. ~~mod_dosevasive~~ permet cependant de monter encore en charge, en dialoguant avec les routeurs afin de lister les adresses à la source d'attaques le plus possible en amont du serveur lui-même. Le fait qu'un attaquant ne soit positionné en liste noire que pendant une période courte permet de s'abstraire d'un grand nombre de complications possibles. Si un attaquant est derrière un proxy, il n'empêche ainsi les autres utilisateurs de son réseau d'accéder à la ressource que pendant un temps très limité.

# Configuration du module

Les directives suivantes sont proposées par le module :

- ~~DOSSiteCount~~ : Le nombre maximal de requêtes qu'une adresse IP source peut réaliser sur le même enfant pendant une unité de temps sans être ajoutée à la liste noire.
- ~~DOSSiteInterval~~ : L'unité de temps (en secondes) évoquée dans la directive DOSSiteCount. La valeur par défaut est d'une seconde.
- ~~DOSPageCount~~ : Le nombre maximal de requêtes qu'une adresse IP source peut réaliser sur la même ressource (même URL) pendant une unité de temps sans être ajoutée à la liste noire.
- ~~DOSPageInterval~~ : L'unité de temps évoquée dans la directive ~~DOSPageCount~~.
- ~~DOSBlockingPeriod~~ : Désigne la durée pendant laquelle tous les accès des adresses IP en liste noire seront refusés et recevront une erreur 403. Par défaut, cette durée est de 10 secondes.
- ~~DOSEmailNotify~~ : Précise une adresse email à laquelle envoyer un courriel lorsqu'une adresse IP est

ajoutée en liste noire. (Attention de bien positionner MAILER dans les sources du module pour utiliser cette directive).
- ~~DOSSystemCommand~~ : Une commande à appeler pour, par exemple, ajouter l'adresse en liste noire sur un routeur.
- ~~DOSWhiteList~~ : Spécifie une adresse IP à ne jamais positionner en liste noire. Il est courant de positionner cette option à 127.0.0.* pour éviter de bloquer nos éventuels propres accès récursifs ou autres robots de référencement. En production, cette option ne devrait jamais être utilisée pour désigner un réseau d'utilisateurs humains légitimes : les mécanismes internes du module garantissent que le trafic légitime ne sera pas bloqué. Cette directive peut être positionnée plusieurs fois avec divers arguments, ceux-ci seront cumulés dans la configuration.

Quelques autres options sont également proposées, pour positionner le répertoire temporaire utilisé par l'outil, ainsi que pour maximiser les performances en optimisant les tables de hash utilisées par l'outil.

# Conclusion

Voici deux modules présentant des approches très différentes mais largement complémentaires pour améliorer la sécurité de serveurs web. Positionnés en proxy entrant à l'entrée d'un réseau hébergeant quelques sites web, ces deux modules peuvent contribuer à assurer une meilleure sécurité, aussi bien en regard d'attaques applicatives visant à prendre la main sur un serveur, que pour protéger ce dernier d'attaques massives visant à l'empêcher de répondre aux requêtes légitimes.

**Liens**


- [1] Sécurisation d'un serveur Apache : Voir Misc 0 ou http://www.miscmag.com/articles/index.php3?page=110
- [2] Site officiel de mod_security : http://www.modsecurity.org/
- [3] Limit et surtout LimitExcept : http://httpd.apache.org/docs-2.0/mod/core.html#limitexcept
- [4] Ivan Ristik, fondateur de la société Thinking Stone : http://www.thinkingstone.com/
- [5] SQL Injection Attacks by Example : http://www.unixwiz.net/techtips/sql-injection.html
- [6] Cross site scripting questions and answers : http://www.cgisecurity.com/articles/xss-faq.shtml
- [7] mod_security stable 1.8 Reference manual : http://www.modsecurity.org/documentation/modsecurity-manual.pdf
- [8] Converted Snort Rules : http://www.modsecurity.org/documentation/converted-snort-rules.html
- [9] Snort : http://www.snort.org/
- [10] Jonathan A. Zdziarski : http://www.nuclearelephant.com/
- [11] mod_dosevasive : http://www.nuclearelephant.com/projects/dosevasive/


~~Retrouvez cet article dans : **Misc 20**~~

Posté par (La rédaction) | Signature : Vincent Deffontaines, Éric Leblond | Article paru dans misc

### Laissez une réponse

Vous devez avoir ouvert une session pour écrire un commentaire.

« Précédent Aller au contenu »
Identifiez-vous
Inscription
S'abonner à UNIX Garden

## Articles de 1ère page


- FUSE, développez vos systèmes de fichiers dans l'espace utilisateur
- SVG – Dessin vectoriel dynamique (1/2)

- Le langage Ada : un peu d'assembleur
- Placement des contrôles dans une fenêtre en C++ à l'aide de wxWidgets
- Entrées/Sorties simples sur USB
- Développement web avancé avec AJAX
- Quelques techniques d'optimisation en C
- Supervision avec OCS Inventory NG et GLPI
- MISC N°41 - Janvier/Février 2009 - Chez votre marchand de journaux
- Le nouveau modèle objet de PHP5



## Catégories

- Administration réseau
- Administration système
- Agenda-Interview
- Audio-vidéo
- Bureautique
- Comprendre
- Distribution
- Embarqué
- Environnement de bureau
- Graphisme
- Jeux
- Matériel
- News

- Programmation
- Réfléchir
- Sécurité
- Utilitaires
- Web

## Articles secondaires

- 30/10/2008
Google Gears : les services de Google offline

Lancé à l'occasion du Google Developer Day 2007 (le 31 mai dernier), Google Gears est une extension open source pour Firefox et Internet Explorer permettant de continuer à accéder à des services et applications Google, même si l'on est déconnecté....

Voir l'article...
7/8/2008
Trois questions à...

Alexis Nikichine, développeur chez IDM, la société qui a conçu l'interface et le moteur de recherche de l'EHM....

Voir l'article...
11/7/2008
Protéger une page avec un mot de passe

En général, le problème n'est pas de protéger une page, mais de protéger le répertoire qui la contient. Avec Apache, vous pouvez mettre un fichier ~~.htaccess~~ dans le répertoire à protéger....

Voir l'article...
6/7/2008
hypermail : Conversion mbox vers HTML

Comment conserver tous vos échanges de mails, ou du moins, tous vos mails reçus depuis des années ? mbox, maildir, texte... les formats ne manquent pas. ...

Voir l'article...
6/7/2008
iozone3 : Benchmark de disque

En fonction de l'utilisation de votre système, et dans bien des cas, les performances des disques et des systèmes de fichiers sont très importantes....

Voir l'article...
1/7/2008
Augmentez le trafic sur votre blog !

Google Blog Search (http://blogsearch.google.fr/) est un moteur de recherche consacré aux blogs, l'un des nombreux services proposés par la célèbre firme californienne....

Voir l'article...

## GNU/Linux Magazine

- Lancement des demi-finales Prologin le 24 janvier
- GNU/Linux Magazine N°112 - Janvier 2009 - Chez votre marchand de journaux
- Édito : GNU/Linux Magazine 112
- Les Éditions Diamond adhèrent à l'APRIL !
- Nouvelle campagne d'adhésion de l'APRIL !

- ## GNU/Linux Pratique

- - Linux Pratique N°51 - Janvier/Février 2009 - Chez votre marchand de journaux
  - Édito : Linux Pratique N°51
  - Linux Pratique HS N°16 - Janvier/Février 2009 - Chez votre marchand de journaux
  - Édito : Linux Pratique HS N°16
  - Linux Pratique HS 16 - Communiqué de presse

- ## MISC Magazine

- - MISC N°41 : La cybercriminalité ...ou quand le net se met au crime organisé - Janvier/Février 2009 - Chez votre marchand de journaux
  - Édito : Misc 41
  - MISC 41 - Communiqué de presse
  - Les Éditions Diamond adhèrent à l'APRIL !
  - Misc HS 2 : Cartes à puce, Découvrez leurs fonctionnalités et leurs limites - Novembre/Décembre 2008 - Chez votre marchand de journaux