TP N°2

Mise en œuvre d’un Reverse Proxy Apache

&

Installation et configuration de ModSecurity 2.1

Objectifs

Ce TP a un double objectif :

- vous permettre d'acquérir les compétences suffisantes pour installer et configurer un **Reverse Proxy** en utilisant Apache 2.2
- vous initier à l'utilisation de **ModSecurity 2.1** pour sécuriser l'accès à des serveurs d'applications hébergés sur un réseau local.

1. Préparation de l'environnement de travail

1.1) Pré-requis

Vous devez disposer sur votre poste de travail du logiciel VMWare Player. Vous pouvez le télécharger à l'adresse suivante : http://www.vmware.com/download/player/ .

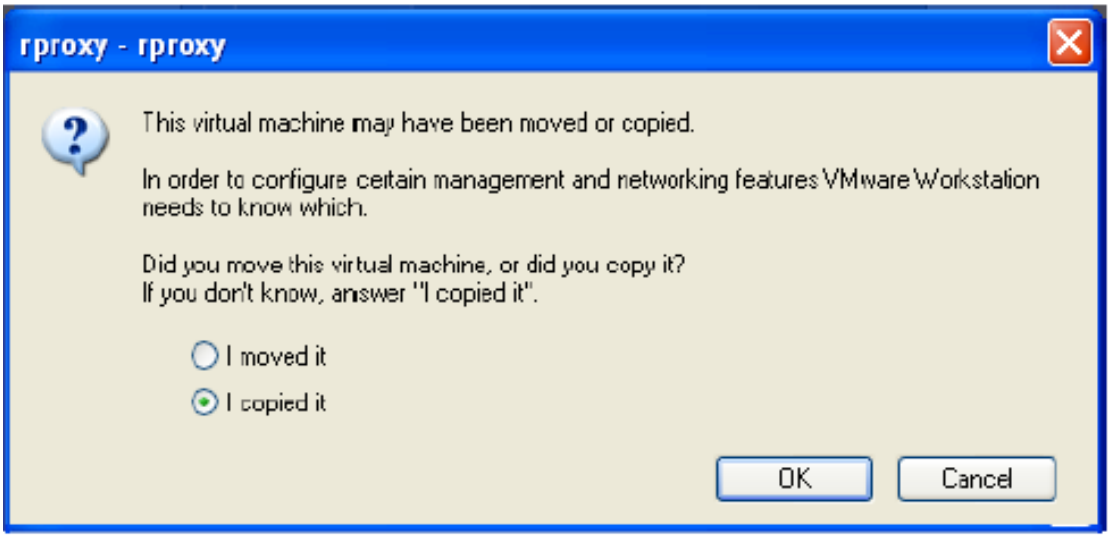
Chaque personne se voit attribuer un triplet d'adresses IP consécutives qui sera utilisé pour le TP.

1.2) Installation et configuration de l'image VMWare du TP

Télécharger et décompresser l'image Reverse_Proxy2.zip. Elle est disponible si nécessaire à l'adresse http://www.ifr88.cnrs-mrs.fr/jtsiars/Reverse_Proxy2.zip**.**

Lancer VMWare Player et sélectionner l'image précédemment décompressée : Reverse_Proxy2. Le système proposé est une Debian Lenny (actuelle version « testing »).

Lors du lancement, le logiciel vous affiche le message suivant :



Sélectionner « I copie dit » et cliquer sur OK.

Lorsque la machine virtuelle est démarrée, vous pouvez vous connecter. Les paramètres sont les suivants :

login : root
password : adf2009

Votre première tâche consiste à configurer les adresses IP utilisées par votre machine virtuelle. Pour cela, chaque personne ou binôme se voit attribuer un triplet d'adresses IP. Les adresses seront de la forme **10.126.100.X**, où X varie de 0 à 2.

Par exemple, la première personne se voit attribuer la plage d'adresses de **10.126.100.100** à **10.126.100.102**, la seconde de **10.126.100.110** à **10.126.100.112**, et ainsi de suite…

Changeons l'adresse IP de la machine hôte (Ulysse). Pour cela nous éditons le contenu du fichier de configuration **/etc/network/interfaces**. Par exemple, si l'adresse IP qui vous a été attribuée est 10.126.100.190 :

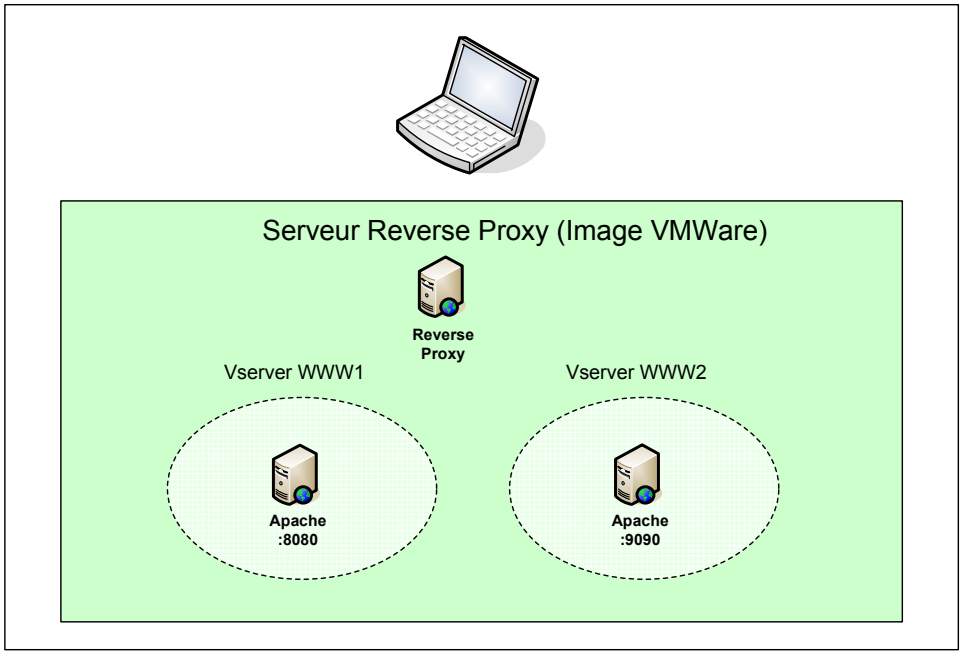
```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet static
        address 10.126.100.190
        netmask 255.255.255.0
        network 10.126.100.0
        broadcast 10.126.100.255
        gateway 10.126.100.250
        # dns-* options are implemented by the resolvconf package, if
installed
        dns-nameservers 194.57.126.30
        dns-search jtsiars.glm
```

Nous relançons ensuite le service réseau pour tenir compte des nouveaux paramètres :

```
ulysse:~# ifdown eth0
ulysse:~# ifup eth0
```

Avant d'aller plus loin, arrêtons-nous un bref instant sur l'architecture physique et logique de l'image virtuelle que nous avons mis à votre disposition :

- une machine hôte (ulysse) héberge un serveur Apache 2.2 qui sera configuré, d'abord en tant que Reverse Proxy, puis comme filtre applicatif grâce au module ModSecurity ;
- cette machine hôte embarque deux serveurs virtuels (www1 et www2) qui seront utilisés pour simuler la présence de serveurs Web « physiques » dans notre réseau local ;
- par commodité (et non par obligation), chacun de ces serveurs Web écoute sur un port différent.



Nous allons également attribuer une adresse IP à chacun des deux serveurs virtuels qui sont utilisés ici pour simuler la présence d'autres machines. Si l'adresse IP 10.126.100.190 a été affectée à votre machine virtuelle VMWare, les adresses **10.126.100.191** et **10.126.100.192** sont respectivement attribuées aux serveurs virtuels www1 et www2.

Nous éditons les fichiers de configuration de chacun des « vservers » :

```
ulysse:~# vi /etc/vservers/www1/interfaces/0/ip
ulysse:~# vi /etc/vservers/www2/interfaces/0/ip
```

Nous pouvons désormais démarrer les deux serveurs virtuels :

```
ulysse:~# vserver www1 start
ulysse:~# vserver www2 start
```

Notre architecture est en place, nous pouvons passer à la configuration du reverse proxy. Pour vérifier que vos serveurs virtuels ont bien démarré, vous pouvez invoquer la commande **vserver-stat** :

```
ulysse:~# vserver-stat
CTX   PROC    VSZ    RSS  userTIME   sysTIME    UPTIME NAME
40000   58 501.8M  10.9M   0m00s36   0m00s26  14m37s73 www1
40001   59 501.7M  10.8M   0m00s64   0m00s29   0m08s16 www2
```

2. Mise en œuvre d'un Reverse Proxy Apache

L'image installée contient déjà les différents binaires et modules Apache nécessaires à la mise en place d'un Reverse Proxy.

2.1) Activation du Reverse Proxy

Mettons en place un reverse proxy depuis la machine hôte (ulysse) vers le serveur Web hébergé sur le serveur virtuel www1. Commençons par activer le module qui active le Reverse Proxy.

```
ulysse:~# a2enmod proxy
Enabling module proxy.
Run '/etc/init.d/apache2 restart' to activate new configuration!
riri:~# /etc/init.d/apache2 restart
Restarting web server: apache2 ... waiting .
```

Nous devons ensuite activer les modules correspondant aux protocoles que nous souhaitons faire transiter par l'intermédiaire du Reverse Proxy. En l'occurrence, il s'agit de **proxy_http**.
NB : La documentation du module **proxy** est disponible à l'adresse suivante : http://httpd.apache.org/docs/2.2/mod/mod_proxy.html .

```
ulysse:~# a2enmod proxy_http
Considering dependency proxy for proxy_http:
Module proxy already enabled
Enabling module proxy_http.
Run '/etc/init.d/apache2 restart' to activate new configuration!
```

Après chaque modification de la configuration du serveur Apache, il faut relancer celui-ci :

```
ulysse:~# /etc/init.d/apache2 reload
Reloading web server config: apache2.
```

2.2) Déclaration des redirections

Passons maintenant à la configuration du ReverseProxy. Depuis la version Apache 2.2, elle se fait par l'intermédiaire du fichier **/etc/apache2/mods-enabled/proxy.conf**. Avant toute chose, désactivons la fonction de relais ouvert via la directive suivante (sur Debian Lenny, cette configuration est activée par défaut) :

```
#turning ProxyRequests on and allowing proxying from all may allow
#spammers to use your proxy to send email.

ProxyRequests Off
```

Puis, nous déclarons les chemins d'accès à nos sites internes. Ensuite, nous préciserons les politiques d'accès au contenu de ces serveurs Web.

Déclarons un relais en direction du serveur Web hébergé sur le port 8080 du serveur virtuel www1. Cette déclaration se fait à l'aide de la directive **ProxyPass**

```
ProxyPass /intranet/ http://10.126.100.191:8080/
```

Déclarons maintenant un second relais sur le serveur Web hébergé sur le port 9090 du serveur virtuel www2. Nous pouvons poser certaines conditions à cette redirection, par exemple, limiter le nombre de connexions simultanées autorisées auprès du serveur www2.

```
ProxyPass /photos/ http://10.126.100.192:9090/ max=20 retry=300
```

Le dernier paramètre indique au serveur Apache qu'il ne doit pas recontacter ce serveur avant 300 secondes si la précédente requête a abouti à une erreur d'indisponibilité.

Vous pouvez tester vos configurations après avoir rechargé la configuration de votre serveur Apache.

2.3) Mise en place d'une politique d'accès

Nous avons rendu accessible notre site Intranet depuis l'extérieur. Bien entendu, nous souhaitons restreindre l'accès aux informations qu'il contient. Suite à la mise en place de la redirection, le serveur www1 ne voit que l'adresse du Reverse Proxy. Il est donc impossible de gérer, à son niveau, des listes de contrôle d'accès en fonction des adresses IP.

En conséquence, ce filtrage doit s'opérer au niveau du Reverse Proxy. Nous pouvons définir des politiques de filtrage différentes en fonction du serveur cible.

```
<Proxy http://10.126.100.191:8080>
      Order deny,allow
      Deny from all
      Allow from 10.234.100.190/255.255.255.255
</Proxy>

<Proxy http://10.126.100.192:9090>
      Order deny,allow
      Deny from all
      Allow from 10.234.100.0/255.255.255.0
</Proxy>
```

Il est également possible de mettre en place des politiques d'accès plus sophistiquées. Par exemple, j'autorise l'accès à mon site Intranet à toutes les machines de mon réseau local et aux machines extérieures à la condition expresse que les utilisateurs de ces dernières s'authentifient auprès de l'annuaire LDAP.

```
<Proxy http://10.126.100.191:8080>
AuthType Basic
      AuthName "Zone Intranet – Accès restreint"
      AuthLDAPURL "ldap://10.126.100.5 :389/ou=accounts,dc=jtsiars,dc=glm"
      AuthBasicProvider ldap
      AuthzLDAPAuthoritative off
      require valid-user
      Order deny,allow
```

```
      Deny from all
      Allow from 10.234.100.190/255.255.255.255
      Satisfy Any
</Proxy>
```

```
riri:~# cp /etc/apache2/sites-available/default-ssl /etc/apache2/sites-available/jtsiars-ssl
riri:~# a2enmod ssl
Enabling module ssl.
See /usr/share/doc/apache2.2-common/README.Debian.gz on how to configure SSL and create self-
signed certificates.
Run '/etc/init.d/apache2 restart' to activate new configuration!
riri:~# /etc/init.d/apache2 restart
```

http://httpd.apache.org/docs/2.2/mod/mod_proxy.html#proxypass

NB : La directive **ProxyVia** indique si le champ **Via :** doit être ajouté à l'entête http, par exemple pour être en mesure d'identifier un serveur parmi une chaîne de mandataires.

3. Intégration de ModSecurity 2.1

3.1) Installation de ModSecurity

Le paquet **libapache2-mod-security** qui implémente le pare-feu applicatif ModSecurity a été retiré de la distribution Debian pour des problèmes de licence. Si ces problèmes sont désormais résolus, le paquet n'a pas encore été réintégré dans la version Debian Lenny pour de simples « raisons logistiques ». En effet, le gel de Lenny est intervenu le 27 Juin 2008 alors que la version du paquet **libapache2-mod-security** libre de droits a été entérinée le 18 Juin 2008. Ce court délai n'a pas permis au responsable officiel du paquet de l'intégrer à l'archive. Cependant, il le diffuse et le maintien sur son site personnel (http://etc.inittab.org/~agi/debian/libapache-mod-security2/README), ce qui nous évitera une compilation.

De manière générale, le site officiel de ModSecurity indique la disponibilité des paquets pour plusieurs distributions (http://www.modsecurity.org).

Nous téléchargeons et installons les deux librairies nécessaires au fonctionnement de ModSecurity :

```
ulysse:~# wget http://etc.inittab.org/~agi/debian/libapache-mod-security2/libapache2-mod-
security2_2.1.5-1_i386.deb
ulysse:~# wget http://etc.inittab.org/~agi/debian/libapache-mod-security2/mod-security2-
common_2.1.5-1_all.deb
ulysse:~# dpkg -i libapache2-mod-security2_2.1.5-1_i386.deb mod-security2-common_2.1.5-
1_all.deb
```

3.2) Configuration de ModSecurity

La documentation se trouve dans le répertoire : **/usr/share/doc/mod-security2-common/**. La procédure de configuration est détaillée dans le fichier **/usr/share/doc/mod-security2-common/examples/rules/README**. Nous la suivons pas à pas.

Commençons à créer le répertoire dans lequel nous allons stocker les règles qui sont fournies par défaut par ModSecurity. Celles-ci sont regroupées dans différents fichiers, selon leur « thématique » ou leur importance.

```
ulysse:~# mkdir /etc/apache2/conf.d/modsecurity
```

```
ulysse:~# cp /usr/share/doc/mod-security2-common/examples/rules/*.conf
/etc/apache2/conf.d/modsecurity/

ulysse:~# ls -l /etc/apache2/conf.d/modsecurity/
total 76
-rw-r--r-- 1 root root 12329 2009-01-21 17:01 modsecurity_crs_10_config.conf
-rw-r--r-- 1 root root  4890 2009-01-21 17:01 modsecurity_crs_20_protocol_violations.conf
-rw-r--r-- 1 root root  2981 2009-01-21 17:01 modsecurity_crs_21_protocol_anomalies.conf
-rw-r--r-- 1 root root  2544 2009-01-21 17:01 modsecurity_crs_23_request_limits.conf
-rw-r--r-- 1 root root  6199 2009-01-21 17:01 modsecurity_crs_30_http_policy.conf
-rw-r--r-- 1 root root  2462 2009-01-21 17:01 modsecurity_crs_35_bad_robots.conf
-rw-r--r-- 1 root root 20108 2009-01-21 17:01 modsecurity_crs_40_generic_attacks.conf
-rw-r--r-- 1 root root  2366 2009-01-21 17:01 modsecurity_crs_45_trojans.conf
-rw-r--r-- 1 root root  6770 2009-01-21 17:01 modsecurity_crs_50_outbound.conf
```

Le fichier **modsecurity_crs_10_config.conf** permet de définir les paramètres de fonctionnement de ModSecurity. Il est possible de préciser, par exemple, quelles parties d'une requête http, entrantes et/ou sortantes, doivent être inspectées (corps, type MIME). La directive **SecServerSignature** permet de modifier la signature renvoyée par le serveur Web. Pour modifier le niveau de logs, notamment lors de vos tests, nous utilisons la directive **SecDebugLogLevel** (la valeur 9 correspond au niveau de logs le plus élevé).

Il est recommandé de documenter les modifications apportées et de conserver une copie de l'original pour pouvoir revenir en arrière en cas de problème.

Nous éditons maintenant le fichier **/etc/apache2/httpd.conf** pour déclarer l'utilisation du module auprès du serveur Apache :

```
# Include modsecurity rules
Include /etc/apache2/conf.d/modsecurity/*.conf
```

Nous avons fini l'intégration de ModSecurity au serveur Apache. Relançons ce dernier pour nous assurer que tout fonctionne bien :

```
ulysse:~# /etc/init.d/apache2 restart
Restarting web server: apache2 Syntax error on line 186 of
/etc/apache2/conf.d/modsecurity/modsecurity_crs_10_config.conf:
ModSecurity: Failed to open the audit log file: /etc/apache2/logs/modsec_audit.log
```

Nous constatons que le module ModSecurity ne peut créer le fichier de traces **modsec_audit_log** déclaré à la ligne 186 du fichier **modsecurity_crs_10_config.conf**. Nous modifions la ligne incriminée et, pour respecter les conventions Debian, nous en profitons pour déplacer le fichier de logs :

```
SecAuditLog /var/log/apache2/modsec_audit.log
```

Nous redémarrons Apache : un problème similaire se produit, cette fois à la ligne 280. Nous modifions également la directive et relançons notre service Web.

```
SecDebugLog             /var/log/apache2/modsec_debug.log
```

Si votre serveur Apache a correctement redémarré, vous pouvez tester votre site et vous assurer qu'il fonctionnent correctement.

Par exemple, essayez de vous consulter votre site en invoquant son adresse IP (ex: http://10.126.100.190)... Notre requête n'aboutit pas. En consultant le fichier de logs de, nous constatons que les règles de base de ModSecurity interdisent les requêtes HTTP basées sur l'adresse numérique de l'hôte.

```
magnum:~# tail -f /var/log/apache2/modsec_debug.log
[24/Jan/2009:16:16:20 +0100] [10.234.224.90/sid#9d19e88][rid#9dc1c98][/][1] Access denied with
code 400 (phase 2). Pattern match "^[\\d\\.]+$" at REQUEST_HEADERS:Host. [id "960017"] [msg
"Host header is a numeric IP address"] [severity "CRITICAL"]
```

Chaque règle possède un numéro permettant de l'identifier. Pour nos tests, essayons de désactiver cette restriction. Elle concerne l'utilisation du protocole HTTP. Par conséquent, elle se trouve donc dans le fichier **modsecurity_crs_21_protocol_anomalies.conf**. Nous la

mettons en commentaire et nous rechargeons la configuration d'Apache. Désormais, le site peut être accédé par son adresse numérique.

Si vous lancez un scanner de vulnérabilités sur votre site, ModSecurity interceptera les requêtes. Certaines de ces règles sont définies dans le fichier **modsecurity_crs_35_bad_robots.conf** :

```
[Sat Jan 24 17:10:34 2009] [error] [client 10.234.224.90] ModSecurity: Access denied with code
404 (phase 2). Pattern match "(?:\\\\b(?:m(?:ozilla\\\\/4\\\\.0
 \\\\(compatible\\\\)|etis)|webtrends security analyzer|pmafind)\\\\b|n(?:-
stealth|sauditor|essus|ikto)|b(?:lack ?widow|rutus|ilbo)|(?:jaascoi|paro)s|webinspe
ct|\\\\.nasl)" at REQUEST_HEADERS:User-Agent. [id "990002"] [msg "Request Indicates a Security
Scanner Scanned the Site"] [severity "CRITICAL"] [hostname "www
.igc.cnrs-mrs.fr"] [uri "//PSUser/PSCOErrPage.htm?errPagePath=/etc/passwd"] [unique_id
"vtphncEy6i8AAAgOBW4AAAAL"]
```

De même, pour coller à l'actualité de la faille Spip révélée en décembre dernier, l'utilisation de ModSecurity permet de nous protéger des injections SQL grâce aux règles définies dans le fichier **modsecurity_crs_40_generic_attacks.conf**. Ainsi, nous disposons d'un délai pour appliquer le correctif de sécurité. Nous parlons alors de correctif virtuel ("virtual patch").

3.3) Ajout de règles supplémentaires

D'autres règles, non activées par défaut, sont mises à disposition dans le répertoire **/usr/share/doc/mod-security2-common/examples/rules/optional_rules/**. Ainsi, le fichier **mod_security_crs_55_marketing.conf** définit des règles qui enregistrent les passages des robots indexeurs de certains moteurs de recherche à des fins purement statistiques.

Vous pouvez également créer vos propres règles. Il est recommandé de ne pas modifier les règles et fichiers fournis par défaut. Par conséquent, vous pouvez créer vos propres fichiers de règles.

3.3.1) Listes blanches

Par exemple, si vous souhaitez mettre en liste blanche l'adresse IP d'une machine, cette nouvelle règle doit être interprétée avant celles fournies par défaut. Vous allez donc créé le fichier **modsecurity_crs_15_custom_rules.conf**. En le nommant ainsi, ce fichier sera interprété immédiatement après le fichier de configuration de ModSecurity (**modsecurity_crs_10_config.conf**).

La règle ci-dessous met la machine 10.126.100.85 sur liste blanche :

```
SecRule REMOTE_ADDR "^10\.126\.100\85$" phase:1,nolog,allow,ctl:ruleEngine=Off
```

Il est possible de désactiver l'application des règles tout en conservant une fonction d'audit pour voir quelles alertes sont levées :

```
SecRule REMOTE_ADDR "^10\.126\.100\85$"
phase:1,nolog,allow,ctl:ruleEngine=DetectionOnly
```

3.3.2) Nouvelles restrictions

Pour ajouter de nouvelles règles de filtrage ou surcharger des signatures qui créent des faux positifs, nous créons cette fois un fichier **modsecurity_crs_60_custom_rules.conf** qui sera interprété en dernier.

Les identifiants numériques de signatures compris entre 1 et 99 999 sont réservés à votre utilisation.

Si vous surchargez une régle existante, vous devez impérativement ajouter à la suite de la nouvelle règle la directive **SecRuleRemoveById** pour indiquer explicitement la signature remplacée. Par exemple, si nous souhaitons modifier la signature **955 004** contre les attaques XSS contenue dans le fichier, nous la réécrivons et nous modifions les paramètres dans notre fichier de configuration :

```
SecRule REQUEST_FILENAME|ARGS|ARGS_NAMES
!REQUEST_HEADERS:Cookie|REQUEST_COOKIES|REQUEST_COOKIES_NAMES|!REQUEST_COOKIES_NAMES:/^Foo$/
"(?:\b(?:(?:type\b\W*?\b(?:text\b\W*?\b(?:j(?:ava)?|ecma|vb)|application\b\W*?\bx-
(?:java|vb))script|c(?:opyparentfolder|reatetextrange)|get(?:special|parent)folder)\b|on(?:(?:
mo(?:use(?:o(?:ver|ut)|down|move|up)|ve)|key(?:press|down|up)|c(?:hange|lick)|s(?:elec|ubmi)t|
(?:un)?load|dragdrop|resize|focus|blur)\b\W*?=|abort\b)|(?:l(?:owsrc\b\W*?\b(?:(?:java|vb)scri
pt|shell)|ivescript)|(?:href|url)\b\W*?\b(?:(?:java|vb)script|shell)|background-
image|mocha):|s(?:(?:tyle\b\W*=.*\bexpression\b\W*|ettimeout\b\W*?)\(|rc\b\W*?\b(?:(?:java|vb)
script|shell|http):)|a(?:ctivexobject\b|lert\b\W*?\())|<(?:(?:body\b.*?\b(?:backgroun|onloa)d|
input\b.*?\btype\b\W*?\bimage|script|meta)\b|!\[cdata\[)|(?:\.(?:(?:execscrip|addimpor)t|(?:fr
omcharcod|cooki)e|innerhtml)|\@import)\b)" \
        "capture,t:htmlEntityDecode,t:lowercase,ctl:auditLogParts=+E,log,auditlog,msg:'Cross-
site Scripting (XSS) Attack. Matched signature <%{TX.0}>',id:'950004',severity:'2'"
SecRule REQUEST_HEADERS|XML:/*|!REQUEST_HEADERS:Referer
"(?:\b(?:(?:type\b\W*?\b(?:text\b\W*?\b(?:j(?:ava)?|ecma|vb)|application\b\W*?\bx-
(?:java|vb))script|c(?:opyparentfolder|reatetextrange)|get(?:special|parent)folder)\b|on(?:(?:
mo(?:use(?:o(?:ver|ut)|down|move|up)|ve)|key(?:press|down|up)|c(?:hange|lick)|s(?:elec|ubmi)t|
(?:un)?load|dragdrop|resize|focus|blur)\b\W*?=|abort\b)|(?:l(?:owsrc\b\W*?\b(?:(?:java|vb)scri
pt|shell)|ivescript)|(?:href|url)\b\W*?\b(?:(?:java|vb)script|shell)|background-
image|mocha):|s(?:(?:tyle\b\W*=.*\bexpression\b\W*|ettimeout\b\W*?)\(|rc\b\W*?\b(?:(?:java|vb)
script|shell|http):)|a(?:ctivexobject\b|lert\b\W*?\())|<(?:(?:body\b.*?\b(?:backgroun|onloa)d|
input\b.*?\btype\b\W*?\bimage|script|meta)\b|!\[cdata\[)|(?:\.(?:(?:execscrip|addimpor)t|(?:fr
omcharcod|cooki)e|innerhtml)|\@import)\b)" \

"capture,t:urlDecodeUni,t:htmlEntityDecode,t:lowercase,ctl:auditLogParts=+E,log,auditlog,msg:'
Cross-site Scripting (XSS) Attack. Matched signature <%{TX.0}>',id:'1',severity:'2'"
```

Cette nouvelle règle signifie que la variable REQUEST_HEADERS de type est Cookie ne doit pas être inspectée. Mais, comme nous ne voulons pas que toutes les variables de ce type échappe à notre filtre, nous précisons que cette exception concerne seulement les cookies nommés “Foo”. Nous attribuons un identifiant numérique à notre règle.

Enfin, nous désactivons la règle fournie par défaut qui posait problème :

```
SecRuleRemoveById 950004
```

3.3.3) Références

http://www.modsecurity.org/documentation/faq.html#d0e400
http://www.modsecurity.org/blog/archives/2007/02/handling_false.html
http://www.modsecurity.org/blog/archives/2007/01/key_advantages.html

3.4) Mises en garde

ModSecurity n'est pas la solution miracle. Ce pare-feu applicatif doit être utilisé avec précaution, surtout dans un environnement de production. Il peut générer des faux positifs.

Il est recommandé de tester au préalable toute nouvelle règle en activant le mode **DetectionOnly** du moteur. Cela se fait par l'intermédiaire de la directive **SecRuleEngine** depuis le fichier de configuration **modsecurity_crs_10_config.conf :**

```
# Turn ModSecurity on ("On"), set to monitoring only
# ("DetectionOnly") or turn off ("Off").

SecRuleEngine DetectionOnly
```

Les signatures/règles utilisées par ModSecurity pour filtrer les requêtes http sont consultables librement par les pirates. Aussi, ils peuvent les étudier à loisir et tenter de trouver des méthodes pour les contourner. Certaines signatures doivent donc être considérées comme un moyen de réduire certaines nuisances, et non pas comme un rempart infranchissable.

Exemple : http://blog.modsecurity.org/2007/02/testing-core-ru.html