



Fiche technique

Apache en tant que reverse proxy

Radosław Pieczonka 

Degré de difficulté



L'ajout d'un reverse proxy permet de bénéficier d'un cloisonnement des flux réseaux et d'un pare-feu applicatif filtrant les requêtes, permettant de renforcer l'architecture globale. Cet article présente comment utiliser Apache en mode reverse proxy.

La plupart des systèmes d'informations proposent des services intranet / extranet, généralement hébergés en DMZ dans les architectures. Directement accessibles depuis d'autres réseaux (dont Internet), ces services peuvent contenir des vulnérabilités inhérentes aux services hébergés (CMS, forums, webmail...), et donc être sujets à des attaques qualifiées et réussies.

Une de ces applications est Microsoft OWA (*Outlook Web Access*), webmail très robuste, qui devient de plus en plus populaire dans la communauté Microsoft Server, devenant ainsi une cible pour les amateurs de failles de sécurité. À partir de ce cas d'étude, nous allons voir comment déployer et contrôler un niveau de sécurité acceptable sur de telles topologies réseaux.

Généralement, les serveurs hébergeant des applications internes sont généralement situés en interne, notamment pour les intranet d'entreprises, ce qui est généralement une très bonne solution tant qu'un accès depuis l'extérieur n'est pas requis. Mais tôt ou tard le besoin d'accéder à ces ressources à partir d'Internet se fera ressentir, notamment pour les parties communicantes comme la messagerie en li-

gne (*webmail*). Quelques scénarios peuvent à ce titre être considérés par les administrateurs réseaux afin de mettre en place une politique de sécurité adéquate.

Scénarios d'architectures

Le premier d'entre-eux consiste à allouer un accès privé virtuel RPV – Réseau Privé Virtuel ou VPN, mais cela requiert une configuration supplémentaire sur les stations clientes, ce que nous éviterons autant que possible dans un premier temps.

Cet article explique...

- Comment utiliser Apache en mode reverse proxy.
- Mettre en place une architecture utilisant un reverse proxy.

Ce qu'il faut savoir...

- Installer un serveur Apache.
- Configurer des hôtes virtuels sur Apache.
- Mettre en place des règles de pare feu en DMZ.

L'hébergement mutualisé

Un second scénario consiste à installer les services en zones d'hébergement mutualisée (Cf. Figure 1), comme le proposent certains fournisseurs de services professionnels. Cette solution a l'avantage de déléguer la maintenance de l'infrastructure (serveurs et services) au fournisseur. Mais elle est toutefois moins flexible et peut entraîner un coût total de possession important. De plus, il peut s'avérer impératif dans certaines situations de stocker les données privées sur le réseau de l'entreprise et non pas sur celui du fournisseur de services.

La mise en DMZ

Un troisième scénario consiste à installer les services dans une zone du réseau de l'entreprise physiquement ou logiquement isolée des autres – zone dite DMZ ou démilitarisée. Située dans le réseau de l'entreprise, celle-ci est soumise à un filtrage s'effectuant généralement via un pare feu IP, élément clef de toute infrastructure de sécurité. Mais, bien que nous ayons plus de contrôle sur cette infrastructure, celle solution n'est pas vraiment différente de la précédente, hormis le fait que les serveurs soient installés dans le système d'information global de l'entreprise. Les requêtes externes provenant d'internet sont passées de la même manière aux serveurs, mais utilisant les adresses et/ou certificats définissant la partie publique de l'infrastructure de l'entreprise, et non plus celle du fournisseur de services.

Le reverse proxy

Quatrième scénario, en poursuivant dans la même direction, et pour apporter une couche de sécurité supplémentaire, un reverse proxy peut être installé sur le pare-feu. Le reverse proxy est une manière plus sophistiquée et délicate à mettre en place, mais très flexible, permettant de multiples et différentes configurations. Dans ce cas d'architecture, les fonctionnalités de routage, pare feu et reverse proxy peuvent être réalisées

par une plate forme basée sur une architecture GNU/Linux ou *BSD, et les services webs (OWA, Intranet, CRM)

placés en DMZ. Dans ce cadre de travail, nous désirons donner accès aux informations depuis l'extérieur tout en

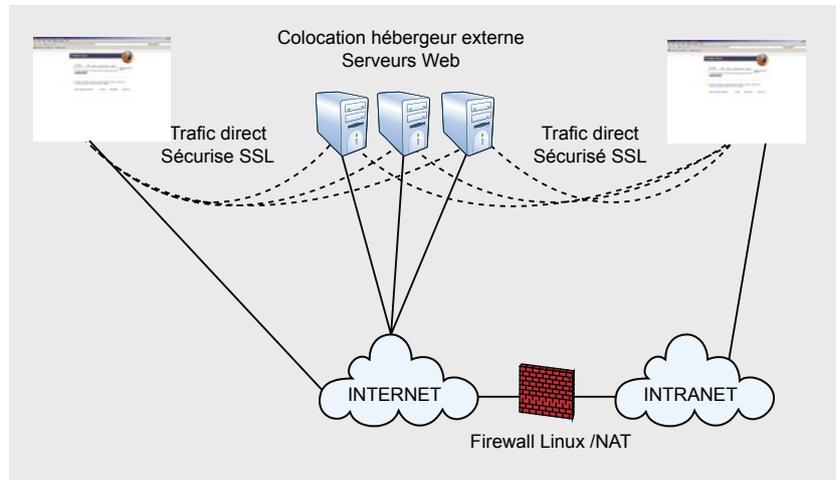


Figure 1. Hébergement des services chez le fournisseur d'accès

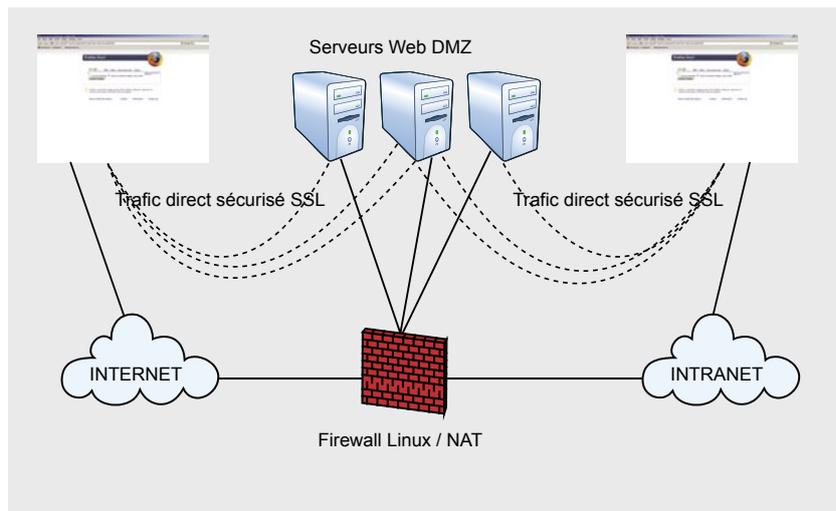


Figure 2. Hébergement des services en DMZ

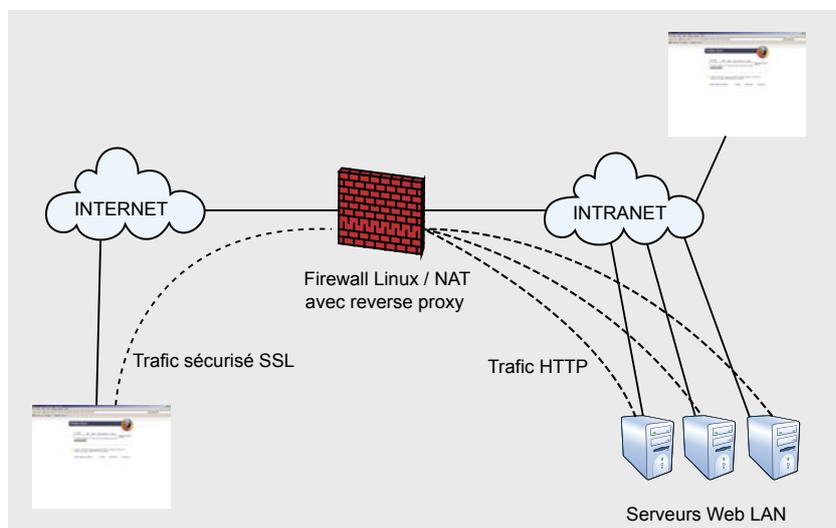


Figure 3. Le reverse proxy mutualisé



assurant les niveaux de sécurité et d'isolement réseau requis.

Le service de reverse proxy peut être délégué et dédié sur un serveur physiquement installé en DMZ. Ceci permet d'assurer le cloisonnement réseau requis et laisser les serveurs web ou intranet dans la zone privée de l'entreprise. C'est le dernier scénario possible et la base de notre architecture de travail.

Conclusion

L'installation d'une architecture utilisant un reverse proxy doit être étudiée en amont avec soin. Un serveur hébergé dans une zone réseau dédiée est une solution flexible et aisément administrable.

La mise en place du reverse proxy

Nous allons maintenant nous intéresser à la configuration du service proprement dit (configuration d'Apache). Premièrement, il faut s'assurer que le serveur web situé en DMZ puisse accéder aux parties attendues (*Outlook Web Access*, la base de connaissance,...), ce qui signifie généralement que nous devons créer des règles ou ACL :

- un jeu de règles d'accès pour que le reverse proxy puisse accéder aux ressources internes,
- un jeu de règles pour que le reverse proxy soit accessible depuis l'extérieur.

Pour publier les informations (provenant de OWA ou de la base CRM), nous utilisons le logiciel Apache doté des modules suivants chargés : *mod_proxy*, *mod_proxy_http*. Le Listing 1 propose une configuration basique des vhosts (hôtes virtuels).

Avec une telle configuration, le domaine interne *owa.lan* sera accessible depuis Internet via l'adresse *intra.company.com*. Nous pouvons dire que cette solution est la manière optimale pour implémenter une telle solution, elle n'inclue pourtant pas d'amélioration concernant la sécurité. Elle ne fait que *mapper* les connexions sur un serveur du réseau

interne. Donc, que pouvons nous faire pour apporter plus de sécurité sur une architecture logicielle basée sur le reverse proxy ? Premièrement, nous devons spécifier les ressources web à publier (i.e. les cibles), et sous quel répertoire web elles seront accessibles via le navigateur :

```
ProxyPass / http://owa.lan/  
ProxyPassReverse / http://owa.lan/
```

en devant spécifier les répertoires qui seront *mappés* par le proxy (Cf. Listing 2).

Comment gérer les connexions chiffrées ?

Grâce à ce changement, l'accès au serveur depuis Internet est limité seulement aux répertoires sélectionnés. Selon l'url demandée, le contenu du site intranet est présenté au client.

Listing 1. Configuration du reverse proxy http

```
<VirtualHost *>  
  ServerAdmin admin@foo.bar  
  ServerName intra.company.com  
  ErrorLog /var/log/apache2/owa.foo.bar-error.log  
  CustomLog /var/log/apache2/owa.foo.bar-access.log common  
  ProxyPass / http://owa.lan/  
  ProxyPassReverse / http://owa.lan/  
  ProxyRequests Off  
  <Proxy *>  
    Order deny,allow  
    Allow from all  
  </Proxy>  
</VirtualHost>
```

Listing 2. Les répertoires

```
ProxyPass /exchange http://owa.lan/exchange  
ProxyPassReverse /exchange http://owa.lan/exchange  
ProxyPass /exchweb http://owa.lan/exchweb  
ProxyPassReverse /exchweb http://owa.lan/exchweb  
ProxyPass /public http://owa.lan/public  
ProxyPassReverse /public http://owa.lan/public  
ProxyPass /knb http://webserver.lan/knowledgebase  
ProxyPassReverse /knb http://webserver.lan/knowledgebase
```

Listing 3. Configuration du reverse proxy https

```
<VirtualHost *:443>  
  ServerAdmin admin@foo.bar  
  ServerName intra.company.com  
  ErrorLog /var/log/apache2/ssl-intra.company.com-error.log  
  CustomLog /var/log/apache2/ssl-intra.company.com-access.log common  
  SSLEngine on  
  SSLCertificateFile /etc/ssl/certs/ssl-cert-intra.company.com.pem  
  SSLCertificateKeyFile /etc/ssl/private/ssl-cert-intra.company.com.key  
  SSLProxyEngine on  
  ProxyPass /exchange https://owa.lan/exchange  
  ProxyPassReverse /exchange https://owa.lan/exchange  
  ProxyPass /exchweb https://owa.lan/exchweb  
  ProxyPassReverse /exchweb https://owa.lan/exchweb  
  ProxyPass /public https://owa.lan/public  
  ProxyPassReverse /public https://owa.lan/public  
  ProxyPass /knb http://webserver.lan/knowledgebase  
  ProxyPassReverse /knb http://webserver.lan/knowledgebase  
  ProxyRequests Off  
  <Proxy *>  
    Order deny,allow  
    Allow from all  
  </Proxy>  
</VirtualHost>
```

Listing 4. Utilisation du `mod_headers`

```
Header unset WWW-Authenticate
Header set WWW-Authenticate "Basic realm=webmail.company.com"
```

Listing 5. Chargement des modules

```
LoadFile /usr/lib/libxml2.so
LoadModule security2_module /usr/lib/apache2/modules/mod_security2.so
LoadModule unique_id_module /usr/lib/apache2/modules/mod_unique_id.so
```

Listing 6. Un fichier vulnérable

```
This site is vulnerable to javascript injection
<?
  if(!isset($_GET['username'])) $_GET['username'] = "no username supplied";
  echo $_GET['username'];
?>
```

Listing 7. Règles `mod_security2`

```
SecRuleEngine On
SecDefaultAction log,auditlog,deny,status:409,phase:2,t:none
SecRule ARGS "<script"
```

Listing 8. Journalisation des requêtes

```
Thu Apr 12 13:09:21 2007] [error] [client xxx.xxx.xxx.xxx] ModSecurity:
  Access denied with code 409 (phase 2). Pattern match "<script" at ARGS:
  username. [hostname "company.com"] [uri "/vulnerable.php?username=
  noob<script>alert(\\\"I%20am%20so%20vulnerable...\\\")</script>"] [
  unique_id "kiwv4Fdi6FkABAVRAMsABBAAA"]
```

Il est important de remarquer que bien que les sites internes soient généralement considérés comme de *confiance*, le trafic internet n'est pas considéré comme tel, c'est pourquoi il est nécessaire de pouvoir gérer les connexions chiffrées.

Pour ceci, nous devons implémenter SSL dans notre exemple en ajoutant les paramètres suivants (Cf. Listing 3).

```
Listen 443
NameVirtualHost *:443
```

Maintenant nous pouvons mettre à jour la configuration des hôtes virtuels présentée en Listing 3, où nous faisons apparaître la directive `SSLProxyEngine`, nous permettant de publier des ressources au travers du protocole https. Si les ressources publiées doivent être mises à disposition des employés de l'entreprise et/ou de tierces parties, il est intéressant d'implémenter une autre couche d'authentification utili-

sateur, via des certificats clients. Le reverse proxy demandera donc un certificat valide devant être fourni par le navigateur, avant d'établir la connexion.

Un autre aspect concernant l'implémentation de la fonctionnalité de `mod_proxy` concerne la répartition de charge et la délégation du chiffrement depuis le serveur applicatif vers le serveur proxy. Lorsque nous publions des ressources http avec le moteur SSL activé sur le reverse proxy public, nous pouvons par exemple desservir plusieurs intranet sous la même racine d'url, et avec le même certificat SSL, ce qui peut être une amélioration significative, tout en permettant de réduire le poste de coût concernant les certificats. Cela peut aussi être une bonne voie pour l'utilisation d'un matériel de type carte accélératrice de chiffrement ssl. Donc cette approche peut servir de socle de travail pour des infrastructures à clefs publiques (PKI).

Configurations particulières

Outre le fait que ceci soit une bonne solution, il se peut que les problèmes surviennent avec des services logiciels ou des configurations serveurs particulières. Par exemple, la publication de l'outil OWA (*Outlook Web Access*) à l'aide de la configuration par défaut du serveur Microsoft Exchange a un inconvénient majeur, situé au niveau de l'authentification NTLM, plus généralement connue en tant que *Windows Integrated Authentication*. En se référant aux publications officielles de Microsoft, l'authentification NTLM ne fonctionnera pas au travers de proxies. Les navigateurs comme Mozilla Firefox, Netscape Navigator, Opera traitent correctement l'information, Microsoft Internet Explorer nécessite de son côté un réglage côté serveur, où l'authentification basée sur NTLM doit être désactivée depuis la console de management d'Exchange sur le serveur IIS. Toutefois, si ce type d'authentification doit être utilisé, le `mod_headers` d'Apache devra alors être utilisé afin de pouvoir réécrire les en-têtes HTTP au niveau du proxy, comme suit (Cf. Listing 4).

Pare feu applicatif

Généralement, le contenu des sites est composé des langages interprétés comme PHP, ASP ou tout autre technologie web similaire, et chaque solution est potentiellement vulnérable à quelques *exploits*. Il n'est pas étonnant de constater qu'une entreprise ne dispose pas des ressources suffisantes pour vérifier chaque partie de code utilisé. quelques fois aussi, le code n'est pas mis à disposition ou délivré, selon les licences logicielles utilisées. Ce qui peut être toléré dans un réseau interne d'entreprise, peut ne pas répondre aux exigences concernant la sécurité lorsque le trafic vient d'Internet. Il est donc impératif de pouvoir filtrer et surveiller sensiblement les accès à notre domaine de publication.

Ceci peut se faire à l'aide du `mod_security2` d'Apache, permettant la mise en place d'un tout autre niveau de



filtrage du protocole. Un des types les plus répandu d'attaque web concerne l'injection SQL. Généralement due à un mauvais contrôle des variables, ce type d'attaque peut être évité en adaptant le code source en conséquence. En imaginant la situation où vous avez une application web importante dont la mise en place n'est pas récente, ne disposant d'aucune maintenance corrective et préventive. Le code de l'application en PHP peut être obsolète, de nouvelles fonctionnalités PHP pouvant être mises à disposition des développeurs, des failles de sécurité de ayant pu apparaître; ceci fait que la mise à jour ou la réécriture de l'application est difficile à envisager dans un cadre opérationnel et financier correspondant à celui de l'entreprise.

D'un autre côté, cette application est cruciale et il s'avère donc impératif de trouver un compromis entre faisabilité et sécurité, afin de ne pas laisser une application *buggée* en ligne. Par exemple, le fait est que si l'application n'est pas protégée contre les injections SQL, une url du type : `http://company.com/buggy.php?username=jonny;DROP%20TABLE%20users`.

Il peut s'avérer dévastatrice sur l'application elle-même. Afin d'éviter de telles *failles*, il est avisé de mettre en place un jeu de règles adéquat à l'aide du `mod_security2` d'Apache. L'installation du `mod_security2` nécessite parfois une compilation du module, ceci est à vérifier auprès de votre éditeur ou de votre distribution gnu/linux préférée. Généralement, une compilation s'avère nécessaire. Vous pouvez voir comment se fait le chargement du module `mod_security2` auprès d'Apache en Listing 5. Après l'installation, nous avons besoin d'activer la fonctionnalité `mod_security2` pour notre reverse proxy, à l'aide de la directive suivante.

```
SecRuleEngine On
SecDefaultAction log,auditlog,deny,
    status:409,phase:2,t:none
```

Ensuite, nous paramétrons le comportement par défaut qui sera adopté par les règles de sécurité comme suit :

Dans cette configuration, le comportement par défaut sera de refuser les requêtes concernées et de les loguer avec un code d'erreur 409. Il est également possible d'outrepasser ce comportement par défaut pour des règles spécifiques, ce que nous verrons plus tard. Il est important d'être informé que `mod_security2` exécute des actions par défaut, afin de réduire

ser ce comportement par défaut pour des règles spécifiques, ce que nous verrons plus tard. Il est important d'être informé que `mod_security2` exécute des actions par défaut, afin de réduire

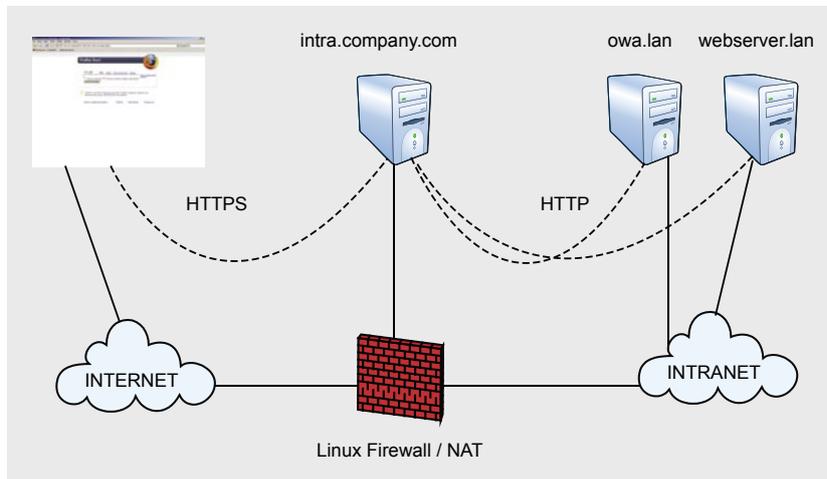


Figure 4. Le reverse proxy dédié



Figure 5. Le site est vulnérable

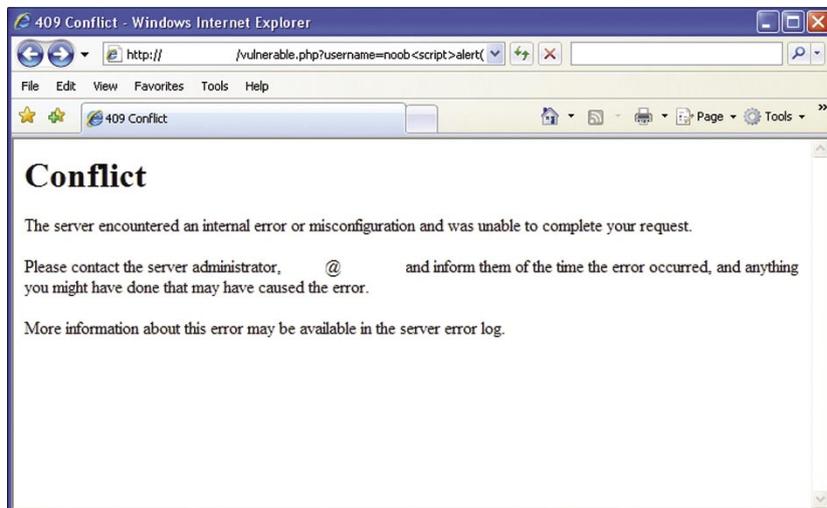


Figure 6. La requête est filtrée

et de contrôler les risques d'attaques. La première consiste à canonicaliser les entrées en remplaçant les slashes multiples (//) les répertoires auto référencés (./), en décodant les codes URL (%00).. La directive la plus communément utilisée avec *mod_security2* est la directive SecRule. Cette directive crée une entrée dans le filtre, et permet d'associer une action à exécuter. Ceci permet de mettre en place un jeu de règles auxquelles sera appliqué le comportement par défaut. C'est un fonctionnement similaire à celui de netfilter, permettant d'associer une action lorsqu'un flux (trame, datagramme, paquet) correspond. *mod_security2* permet aussi de détecter des signatures dans le code source, ceci permet de couvrir la plupart des problèmes de sécurité connus dans les application web.

Plus communément connu sous le nom de *core rules*, l'utilisation et l'analyse de ce jeu de règle disponible par défaut est fortement recommandée, permettant de se pré-munir des attaques les plus connues et utilisées. Dans ce jeu de règles *core rules*, chaque directive peut recevoir 2 ou 3 attributs. Le premier déclare où chercher la signature, le second est la signature elle-même, le troisième pouvant être une option. Une action pourra alors être associée, sinon le comportement par défaut sera adopté. Pour illustrer comment le *mod_security2* peut protéger la publication des informations web, nous pouvons créer un script PHP *buggé* qui sera installé sur un serveur web sommairement configuré, le rendant vulnérable : contenu du fichier *vulnerable.php* (Cf. Listing 6). Lorsque le navigateur web envoie la requête, nous pouvons

exploiter la vulnérabilité d'injection de script en soumettant le code du script en tant que nom d'utilisateur, comme suivant :

```
like ?username=jonny<script>
    alert("I am so vulnerable...")
</script>
```

et comme illustré en Figure 5. Nous pouvons résoudre ce problème sur l'application ou le serveur web lui-même, mais le reverse proxy peut être l'élément clef pour intervenir en étant le *hub* de filtrage stratégique dans une infrastructure réseau/sécurité. Nous allons voir comment intervenir sur ce type de trafic à partir du reverse proxy filtrant. Si nous établissons une architecture réseau basée sur Apache en reverse proxy et si nous activons le *mod_security2*, nous pouvons simplement bloquer les requêtes contenant la chaîne `<script>`, et journaliser ces tentatives d'accès avec un code d'erreur spécifique, de la manière suivante (Cf. Listing 7). Après une telle modification, l'attaquant recevra le code d'erreur 409 et toutes les tentatives d'accès seront journalisées de la manière suivante (Cf. Listing 8).

Configuré de cette manière, le site est protégé par le reverse proxy filtrant. En travaillant avec *mod_security2*, il est important de noter que la mise en œuvre d'une politique de filtrage de flux doit être étudiée et testée avec soin, afin de ne pas polluer et déranger l'utilisation normale de ressources mises à disposition des utilisateurs.

Conclusion

Comme présenté dans cet article, le reverse proxy peut être utilisé afin d'améliorer la sécurité concernant la publication des informations de manière significative. Il peut s'avérer stratégique pour respecter le cloisonnement physique et logique des réseaux d'entreprise. Et cela peut être mise en place grâce à des technologies basées sur des logiciels open source. La solution *Apache2/mod_proxy/mod_security2* est une alternative à des solutions de type ISA(c) proxy server. ●

À propos de l'auteur

Radosław Pieczonka travaille depuis un an pour IFRResearch Pologne, filiale Recherche et Développement de WALLIX, en tant qu'Intégrateur et Analyste Système et Réseau. Pour contacter l'auteur :

radek.pieczonka@wallix.com

<http://www.wallix.com>

<http://www.ifresearch.pl>



Vous allez y trouver :

- matériaux complémentaires aux articles - listings, outils indispensables
- les articles les plus intéressants à télécharger

www.hakin9.org