Squid

1) Présentation

Squid est un logiciel permettant la réalisation d'un cache pour les clients web. Squid peut aussi jouer le rôle de filtre http.

Squid est un produit disponible sur un grand nombre de plates-formes:

- ➔ Linux
- ➔ FreeBSD, NetBSD
- ➔ OSF/ Dec Unix
- ➔ AIX
- ➔ HP-UX
- ➔ Windows NT

Squid est un logiciel libre protégé par la GPL (Gnu Public Licence).
Son url est http://www.squidguard.org
L'adresse mail de référence est : squidguard@ost.eltele.no

Ses auteurs sont :

- Päl Baltzersen : pal.baltzersen@ost.eltele.no
- Lars-Erik Häland : leh@nimrod.no

2) Le proxy cache

Squid est un cache proxy :

- **Proxy :** un agent qui agit pour un autre
- **Cache :** un espace où l'on stocke les informations les plus usitées

Squid agit comme un agent qui reçoit des requêtes de clients, les transmet au serveur Internet concerné et stocke l'information retournée par le serveur Internet. Si l'information est demandée plusieurs fois, alors le contenu stocké sera retourné aux clients, ce qui réduit la bande passante utilisée et accélère les opérations.

La majorité des firewalls disposent d'agents Internet; toutefois ceux-ci ont rarement la fonctionnalité de cache ou ne supportent pas l'intégralité des fonctionnalités de Squid.

Squid se distingue par un grand nombre de fonctionnalités avancées :

- ➔ Support de nombreux protocoles (les protocoles de transfert de données utilisées sur Internet comme http, Gopher, ftp et Wais)
- ➔ Possibilité d'interconnexions entre différents serveurs proxys : pour de gros réseaux il peut être nécessaire d'installer plusieurs proxys caches pour répartir la charge. Dans ce cas, il est intéressant de les relier et de les faire coopérer pour une meilleure efficacité.
- ➔ Contrôle d'accès : Squid est capable de filtrer certains sites Internet ou certaines URL ou de restreindre l'accès à certains clients.

Squid est en outre un produit hautement modulaire architecturé autour d'un programme principal chargé de la gestion du cache des données nommé *squid,* d'un serveur de cache de noms de domaine *dnsserver.*
Par ailleurs, d'autres programmes se chargent de la réécriture des url, de l'authentification et de sa gestion. Cette architecture modulaire permet le développement de modules additionnels permettant de compléter les fonctionnalités de Squid.

3) Limitations

Squid n'est pas un Firewall. Il peut limiter les possibilités des clients mais il ne protège pas l'accès aux personnes extérieures au réseau.
La gestion du pare feu sous Linux est réalisée par ipchains pour les noyau 2.2 ou iptables pour les noyaux 2.4.
Squid ne supporte pas tous les protocoles. Il est ainsi incapable de gérer les protocoles de news, real audio et de vidéo conférences. Il est nécessaire d'utiliser d'autres caches applicatifs.

4) Protocole supportés

Squid supporte les protocoles suivants :

- HTTP: le protocole Web
- FTP : le protocole de transfert de fichiers
- Gopher : un protocole proche de http inusité aujourd'hui
- SSL : utilisé pour les transactions sécurisées
- WAIS : protocole inusité
- ICP : un protocole généraliste permettant la communication entre serveurs de cache
- HCTP : un protocole de cache orienté http

5) Installation

L'installation de Squid est relativement simple et respecte le standard « configure ; make ; make install ».
Afin de mettre en place un serveur opérationnel, il est important de connaître les pré requis de Squid.

a) Prérequis

Il est important pour obtenir un cache efficace de lui offrir une quantité importante de mémoire vive et de disque

- 128 Mo de ram
- 8 Go de disque.

A l'inverse, les besoins CPU sont bien avec moindre.
Ainsi, de nombreux pentium de serveurs SQUID fonctionnent première génération.
encore des processeurs de première génération.

b) Installation

Il existe deux méthodes pour installer SQUID

- Installation de binaires à partir de paquetage
- Installation à partir des sources

Installation à partir des binaires

Le logiciel squid est fourni avec la plupart des distributions commerciales :

- Red Hat
- Mandrake
- SuSE
- Caldera.

Dans tous les autres cas, des paquetages binaires (au format rpm) de Squid peuvent être trouvés sur le site www.rpmfind.net. Une fois les binaires à disposition, il ne reste plus qu'à les installer en utilisant la commande rpm ou tout autre gestionnaire de paquetages (kpackage ou gnorpm) :

```
rpm -ivh squid-XXX.rpm
```

Installation à partir des sources

Les sources de Squid sont disponibles sur le site www.squid - cache.org.
Une fois le fichier rapatrié, tapez les commandes suivantes :

```
tar zxvf squid-xxx.tar.gz
cd squid-xxx
```

```
./configure
make
make install
```

Quelques options du script configure :

--prefix = **path :** spécification du chemin d'installation
--enable-htpc : active le port htpc (utile pour faire fonctionner plusieurs serveurs de cache ensemble)
--enable-snmp : permet de générer un démon snmp.

Un exemple de configuration de la compilation :

```
/configure --prefix= /opt/squid --enable-snmp --enable-htpc
```

6) Configuration

a) Généralités :

Pour une installation « source », il est important de créer un user et un groupe squid. Celui – ci permettra l'éxécution du programme squid sous un autre user que « root ».
Le fichier de configuration est /etc/squid/squid.conf. Le fichier par défaut est complet et les directives sont commentées entre les lignes (#)
Il est conseillé de faire une copie du fichier original est d'utiliser son propre fichier squid.conf.

b) Directives :

voici quelques directives incontournables :

- utilisateur :
  cache_effective_user et cache_effective_group définissent sous quel user/group fonctionne squid
  ```
  cache_effective_user squid
  cache_effective_group  squid
  ```
- port :
  permet de spécifier le port de connection des clients. Par convention, le port est 3128 mais on peut prendre aussi le 8080.
  ```
  http_port 3128
  ```
- répertoire de cache :
  indique ou seront stockés les objets « cachées ». Attention, ce répertoire doit apparetenir à l'utilisateur définit ds la directive « cache_effective ». Il faut également indiquer le type d'organisation de fichier et l'espace disque alloué.
  Voici la syntaxe :
  ```
  cache_dir <type disk> <répertoire> <taille en Mo> <nb rép><nb sous rép>
  ```
  En général, le type est ufs et il faut créer ces répertoires avant l'utilisation par la commande squid -z.

```
cache_dir ufs /www-cache 1024 16 256
```

- adresse administrateur :
  indique l'adresse mél de l'admin du programme. En cas de plantrage par ex un courrier lui sera envoyé.
  ```
  cache_mgr slismaster@ac-nantes.fr
  ```
- taille maximale des objets :
  Squid stocke les objets dans son cache selon une taille définit par des limtes inférieures et supérieures.
  ```
  minimum_object_size 0 kb
  maximum_object_size 1024 kb
  ```
- Cache DNS :
  Squid effectue des requêtes DNS qui sont « blocantes ». Il démarre ainsi un certains nombre de processus pour répondre à ces requêtes. On spécifie leur nombre avec la directive dns_children. Si cette requête a réussie, elle est placée en cache pendant le temps précisée ds la directive positive_dns. Dans le cas contraire, le temps sera celui de negative_dns.
  ```
  dns_children 10
  positive_dns_ttl 24 hours
  negative_dns_ttl 5 minutes
  ```
- validité et rafraîchissement des objets :
  Deux directives permettent d'influencer le comportement du mandataire quant à la façon dont il jugera utile de recharger la page demandée depuis le *Web* plutôt que de la fournir directement au client.

  - La directive reload_into_ims permet, lorsque sa valeur on de passer directement la page à un client sans la recharger même si celui-ci le demande explicitement, à la condition toutefois que la page dans le cache soit à jour. Le mandataire vérifie la validité de cette condition en passant une requête au serveur distant avec l'argument if-modified-since.
  L'utilisation de cette fonctionnalité constitue une violation du protocole http.
  ```
  reload_into_ims on
  ```

  - La directive refresh_pattern permet un contrôle beaucoup plus fin de la validité des objets cachés. Son utilisation requiert la compréhension de l'algorithme de gestion du rafraichissement de Squid.
  ```
  refresh_pattern [-i] <url_regexp> MIN_AGE PERCENT MAX_AGE <options>
  ```
  Les lignes ci-dessous sont les valeurs par défaut :

  ```
  refresh_pattern         ^ftp:           1440    20%     10080
  refresh_pattern         ^gopher:        1440    0%      1440
  refresh_pattern         .               0       20%     4320
  ```

  voir paragraphe suivant pour l'algorithme

c) L'algorithme de cache :

Principe de base de l'algorithme de gestion du rafraichissement

Lorsqu'il reçoit une requête pour un objet, le mandataire décide du comportement à adopter en fonction de l'état de l'objet dans son cache. Un objet déclaré « frais » est fourni directement au client. S'il est « périmé », alors une requête if-modified-since est envoyée au serveur distant.

Pour déterminer la « fraîcheur » d'un objet, Squid utilise un certain nombre de variables.

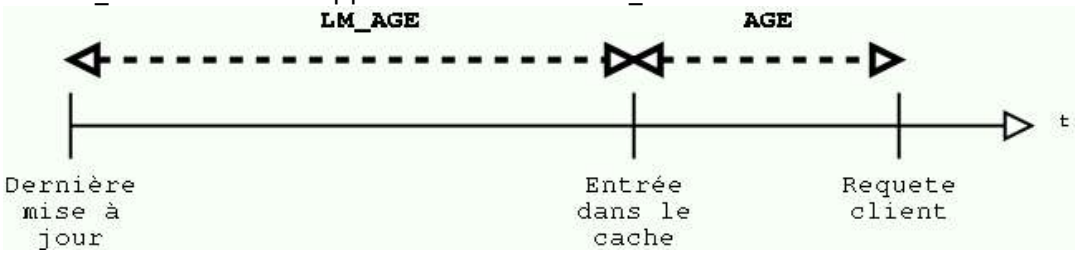
Variables utilisées par l'algorithme

Squid utilise trois types de variables dans sa détermination : des variables correspondant à l'objet enregistré, une variable fournie par le client et des variables du fichier de configuration.

Variables associées à l'enregistrement

Ces variables sont calculées d'après les données relatives à l'enregistrement de l'objet dans le cache.

- AGE correspond au temps écoulé depuis son entrée dans le cache.
- LM_AGE correspond au temps écoulé entre la dernière modification de l'objet sur le serveur distant et son entrée dans le cache. Une grande valeur de LM indique que l'objet était déjà âgé lors de son entrée dans le cache. C'est un indice du rafraichissement de l'objet sur le serveur.
- LM_FACTOR est le rapport entre AGE et LM_AGE.



Plus ce facteur est grand, plus l'objet a de chances d'être périmé.

- EXPIRES est la valeur éventuellement fournie par le serveur[6] au moment de l'entrée de l'objet dans le cache. Comme son nom l'indique, cette variable contient la date d'expiration de l'objet.

Variable fournie par le client

Le mandataire tient également compte de la variable de contrôle de cache CLIENT_MAX_AGE éventuellement fournie par le client, s'il utilise la version 1.1 d'http. Cette variable indique l'âge maximal de l'objet accepté par le client.

Variables du fichier de configuration

C'est la directive refresh_pattern qui permet de fixer les variables correspondant à l'algorithme de gestion du rafraichissement. Sa syntaxe est la suivante :

```
refresh_pattern [-i] <url_regexp> MIN_AGE PERCENT MAX_AGE <options>
```

Le drapeau **-i** permet de ne pas tenir compte de la casse des caractères dans l'expression régulière de description de l'url.

Les variables qui permettent de régler le comportement du cache sont MAX_AGE, PERCENT et MIN_AGE.

Il peut y avoir plusieurs directives refresh_pattern dans le fichier de configuration. Ces lignes sont lues à chaque requête, la première correspondant à la requête[8] fixant les valeurs des avariables.

Les lignes ci-dessous sont les valeurs par défaut :

```
refresh_pattern         ^ftp:           1440    20%     10080
refresh_pattern         ^gopher:        1440    0%      1440
refresh_pattern         .               0       20%     4320
```

Les valeurs de MAX_AGE, PERCENT et MIN_AGE pour les requêtes http sont donc respectivement 0, 20% et 4320 (unités de temps en minutes).

Algorithme

La variable MAX_AGE éventuellement fournie par le client prévaut sur celle du fichier de configuration. Par contre, MIN_AGE du fichier de configuration est prioritaire devant la variable EXPIRES éventuellement fourni par le serveur.

Une fois les variables nécessaires déterminées, Squid observe les règles suivantes pour déterminer la fraîcheur de l'objet dans le cache :

- Si MAX_AGE est défini et que AGE y est supérieur, l'objet est déclaré « périmé » ;
- Si AGE est inférieur ou égal à MIN_AGE, l'objet est déclaré « frais » ;
- Si EXPIRES est défini et dépassé, l'objet est déclaré « périmé » ;
- Si EXPIRES est défini mais pas encore dépassé, l'objet est déclaré « frais » ;
- Si AGE est supérieur à MAX_AGE, l'objet est déclaré « périmé » ;
- Si LM_FACTOR est inférieur à PERCENT, l'objet est déclaré « frais » ;
- Aucune autre règle n'ayant abouti dans la détermination de la fraîcheur de l'objet, celui-ci est déclaré « périmé ».

La ligne suivante :

```
refresh_pattern ^http 1440 10% 10080
```

permet de charger les documents Web une seule fois par jour, avec un taux de rafraichissement plus faible que la norme et qu'il n'aient aucune limite d'âge dans le cache.

d) les logs
Importants en consultation, il permettent un audit des états de Squid mais aussi une journalisations des accès par les clients. Ils sont en général stockés dans : /var/log/squid

7) Sécurité

La sécurité est l'une des fonctionnalités les plus intéressantes de Squid. Celui-ci dispose, en effet, de fonctions de filtres sur des protocoles non Supportés comme http ou ftp par les pare - feu de filtres à paquets.

a) Limitation :

Il est possible de restreindre l'utilisation de Squid à certaines personnes.

Ces limitations sont écrites à partir d'ACL (Access Control List) et d'ACO (Access Control Operator).
On construit un ensemble de règles nommées puis celles-ci pourront être utiliséesdans des directives. La syntaxe des ACL est la suivante :

```
Acl nom acl acl_type ...
```

Les différents types d'acl sont :

src : cette acl est basée sur l'adresse IP d'un client
dst : cette acl est basée sur l'adresse IP demandée dans une requête
srcdomain : cette acl est basée sur le nom de domaine du client
dstdomain : cette acl est basée sur le nom de domaine du serveur contacté
port : cette acl se base sur le port de destination
proto : cette acl se base sur le protocole

Il existe bien d'autres types d'ACL qui permettent d'élaborer des règles extrêmement fines.
Quelques exemples d' ACL :

```
acl src localhost src 127.0.0.1/255.255.255.0
acl src local src 192.168.1.0/255.255.255.0
acl all src 0.0.0.0/0.0.0.0
```

Ensuite, nous pouvons utiliser ces règles sur les accès. Il existe différents types d'accès : http access et icp access.
Leurs syntaxes : http access allow deny acl_name icp access allowldeny acl_name Quelques exemples de règles:

```
http_access allow localhost
http_access allow  src_local
```

```
icp_a ccess deny all
```

b) Désactivation du cache

Il peut être intéressant de désactiver la fonction de cache pour certains sites (si les sites sont hébergés localement,le cache n'offrira pas *d'améliorations de performances). On utilise alors la directive always direct avec la propriété allow ou deny sur une ACL :*

```
acl adresses_locales dst 192.168.0.0/255.255.0.0
always direct allow addresses_locales
```

De plus, il n'est pas nécessaire de sauvegarder ces données avec Squid.
On utilise à cette fin la directive no cache avec l'option deny et le nom de l'ACL concernée*:*

```
No cache deny adresses_locales
```

c) horaires d'ouverture :

Le type de contrôle time prend deux arguments, une lettre correspondant au jour de la semaine sous la forme d'une lettre[4] , et une fourchette horaire, sous la forme suivante :

```
<h1:m1-h2:m2>, par exemple 08:30-18:45
```

Ainsi, la ligne suivante :

```
acl heure_ouverture time M 08:30-18:45 \
                         T 08:30-18:45 \
                         W 08:30-18:45 \
                         H 08:30-18:45 \
                         F 08:30-18:45
```

définit un contrôle d'accés pendant une semaine de travail standard pour un établissement scolaire.
On peut noter également que les arguments de base peuvent être répétés pour préciser le contrôle d'accés.

8) Collaboration de caches

Dans le cadre de grands réseaux d'entreprise, il peut être utile d'installer plusieurs proxys caches. Toutefois, afin de ne pas fragmenter les caches, il est possible de relier différents serveurs caches. Cette technique est employé sur nos architectures établissements qui mettent en jeu le couple Amon / SLIS.
Le SLIS est configuré pour avoir comme proxy « parent » le serveur Amon lui même

service proxy de tête.

Il est aussi possible de mettre en place une répartition de charge[1] pour obtenir une solution optimale.

1 Répartition de charge : mise en commun de plusieurs machines pour en obtenir une virtuelle de puissance cumulée. D'un point de vue utilisateur, il n'existe qu'une seule machine.

Afin d'obtenir une répartition de charge, on définit un nom de machine identique aux deux serveurs de cache d'adresse IP IP1 et IP2 de nom serveurl et serveur2. Il suffit d'ajouter dans le fichier de zone de Bind :

```
proxy IN A IP1
proxy IN A IP2
```

Dans la configuration des navigateurs,il suffit de donner dans la configuration du proxy le nom de machine *proxy.* Le serveur de nom (Bind) renverra l'une ou l'autre des adresses IP.

Dans le fichier de configuration du serveurl, on ajoutera:

```
cache_peer serveur2 sibling 3130 proxy-only
acl src serveur2 src IP2/255.255.255.255
icp-access allow src_serveurl
```

Et dans le fichier de configuration du serveur2, on ajoutera :

```
cache_peer serveurl sibling 3130 proxy-only
acl src serveurl src IP1/255.255.255.255
icp_access allow src_serveurl
```

Notons que le port 3130 est le port ICP.

SquidGuard

1) présentation :

Squid-guard est un plugin de Squid. Il offre les possibilités suivantes :

- Filtrage
- Redirection
- Authentification

Il s'agit en outre d'un logiciel libre (sous licence GPL), portable (disponible sur la majorité des systèmes Unix). On pourra utiliser Squid pour ces tâches :

- Limiter l'accès web à quelques utilisateurs
- Limiter l'accès à certains sites Web
- Bloquer l'accès à certaines url
- Envoyer les utilisateurs non enregistrés sur Squid sur un formulaire d'enregistrement
- Rediriger les bannières sur des fichiers GIF vides.
- Avoir des règles basées sur des dates ou sur des groupes

2) Installation :

Avant d'installer SquidGuard, il est nécessaire d'installer Berkeley DB 2.

On récupère les sources de SquidGuard sur le site officiel www.squid-quard.org. On décompresse les sources avec la commande *tar* et on va ensuite dans le répertoire créé pour y effectuer les commandes classiques :

```
./configure
make
make install
```

3) Configuration :

La configuration de squidguard est réalisée dans le fichier squidguard.conf. Il est en général installé dans le répertoire /usr/local/squiguard/squidguard.conf
Pour expliquer configurations types. le fonctionnement, nous allons présenter quelques configurations types

a) Configuration élémentaire

Par défaut, on écrit le fichier de configuration suivant qui autorise tout le monde à

accéder à l'ensemble des sites :

```
logdir /usr/local/squidGuard/log
      acl {
      default {
             pass all
      }
}
```

b) Configuration permettant d'éliminer des domaines

La configuration suivante permet de rendre inaccessible des sites Internet en les sélectionnant dans un fichier :

```
logdir  /usr/local/squidGuard/log
         dbhome /usr/local/squidGuard/db
                dest local {
                domainlist local/domains
               }
               acl {
                 default {
                      pass local none
                      redirect http://localhost/refu.htm
                 }
              }
```

Dans le fichier "/usr/ local /squidGuard/db/ local /domains", on indiquera la liste des sites refusés : par exemple : microsoft.com ;-)

c) Configuration autorisant certains clients à utiliser Squid

```
logdir /usr/local/squidGuard/log
dbhome /usr/local/squidGuard/db
src privileged {
             ip   10.0.0.1 10.0.0.73 10.0.0.233
             ip   10.0.0.10-10.0.0.20
             ip   10.0.1.32/27
             ip   10.0.2.0/255.255.255.0

             domain foo.bar
 }
acl {
   privileged {
          pass all
   }
   default {
          pass none
          redirect http://info.foo.bar/refu.htm
  }
}
```

Attention, il est important d'activer dans Squid l'option *log fqdn on.* Ceci est utile pour définir des domaines (dans notre cas, le domaine foo.bar).

d) Sélection des domaines et des url non authorisés

On interdit l'accès aux domaines cités dans le fichier games/domains et les url citées dans le fichier games/urls. Voici le fichier squidguard.conf :

```
logdir /usr/local/squidGuard/log
dbhome /usr/local/squidGuard/db
      dest games {
      domainlist games/domains
            urllist  games/urls
            }
acl {
    default {
    pass !porn all
    redirect http://localhost/refu.htm
    }
}
```

Le fichier domains est semblable aux exemples ci-dessus et le fichier urls :

foo.com/games
bar.com/img/games

e) Autoriser l'accès à certains sites à certains clients

Exemple :

```
logdir /usr/local/squidGuard/log
dbhome /usr/local/squidGuard/db
src direction {
      10.0.0.0/24
      user foo bar
 }
src administratif {
       ip   10.0.0.0/22
 }
dest jeux {
      domainlist jeux/domains
      urllist     jeux/urls
 }
acl {
      direction {
            pass all
      }
      administratif {
```

```
            pass !jeux all
      }
      default {
            pass none
            redirect http://info.foo.bar/refu.htm
      }
 }
```

Cette configuration permet d'autoriser les membres de direction à se connecter où bon leur semblent et d'empêcher aux membres administratifs l'accès aux sites et url de jeux précisés dans les fichiers :
/usr/local/squidguard/db/jeux/domains et /usr/local/squid/db/jeux/urls.

f) Configuration des accès selon les dates

```
logdir /usr/local/squidGuard/log
dbhome /usr/local/squidGuard/db
time leisure-time {
               weekly * 00:00-08:00 17:00-24:00 # night and evening
               weekly fridays 16:00-17:00 # weekend
               weekly saturdays sundays           # weekend
 }

src direction {
     ip  10.0.0.0/24
     user foo bar
 }
src administratif {
     ip  10.0.0.0/22
 }
dest jeux {
     domainlist  jeux/domains
     urllist      jeux/urls
          redirect      302:http://info.foo.bar/images/blocked.gif
# redirect matches to this URL
 }
acl {
      direction within leisure-time {
            pass all
      } else {
            pass !in-addr !jeux all
      }
      administratif {
            pass !jeux all
      }
      default {
            pass none
            redirect http://info.foo.bar/refu.htm
      }
}
```

g) Intégration de squidguard dans Squid

Cette opération est réalisée de manière extrêmement simple, il suffit d'éditer le fichier de configuration de squid nommé squid.conf et de rajouter :

```
redirect_program /prefix/bin/squidGuard -c
/wherever/squidGuard.conf
redirect_children 4(*)
```

(*) Voir le paragraphe suivant

Les redirecteurs :

Redirecteurs : *un redirecteur est un programme qui "s'attache" à Squid et qui permet une réécriture des URLs avant de les passer au cache proprement dit. Cela permet de modifier les Urls*

- soit pour restreindre les accès (Urls à caractère pornographique, ludique, etc...)
- soit pour faire disparaître les publicités (et donc accélérer considérablement le trafic, les bandeaux étant généralement régénérés à chaque appel)
- soit pour diriger l'utilisateur vers un miroir local (par exemple les téléchargements de Netscape Navigator)

Pour les mettre en place, il suffit de rechercher dans le squid.conf la ligne redirector :

```
redirect_program /usr/local/squidGuard/bin/squidGuard
```

et de préciser le nombre de fils que l'on souhaite :

```
redirect_children 20
```

Attention, mettez en assez, pour que le processus principal ne soit pas bloqué en attente d'un process redirecteur libre. Inutile non plus d'en mettre trop (charge mémoire).
Les redirecteurs font quasiment tous, au départ, de 800 Ko à 1600 Ko.
De toute manière Squid en limite le nombre à 32.

Références :

Site de SquidGuard.conf :
http://www.squidguard.org

site de configuration de Squidguard :
http://www.squidguard.org/config/

docs intéressante sur Squid :
http://linux.crdp.ac-caen.fr/Docs/Squid/t1.html