

TP Squid



I- Présentation Squid

Squid, est un logiciel libre, open source qui a pour rôle de proxy, c'est une solution de proxy cache. Squid est en perpétuelle évolution et s'est adapté au fil du temps aux différentes versions du protocole HTTP. Il est distribué sous licence GNU GPL (General Public License) et disponible en libre téléchargement à partir de son site Internet officiel : <http://www.squid-cache.org> Squid supporte aujourd'hui pratiquement tous les standards Internet et utilise les principaux protocoles comme HTTP, HTTPS et FTP. Il permet aussi l'interconnexion de serveur cache. En plus de stocker en mémoire cache les métadonnées et données les plus régulièrement appelées par les utilisateurs, il garde aussi en mémoire les requêtes DNS afin d'optimiser ses temps de réponse.

Le proxy a le rôle d'intermédiaire entre le client et internet. Le proxy a plusieurs rôle :

- Le rôle de mandataire : « le proxy va effectuer toutes les requêtes sur le web à notre place. »
- De cash : « il garde en cash toutes les pages consultées, cela est un gain de temps car il n'aura pas besoin d'aller chercher ces informations sur le web. »
- Et de filtrage : « le filtre peut se faire sur des URL, horaire etc »

II- Installation

Nous commençons tout d'abord par mettre à jour avec la commande apt update et apt upgrade :

```
root@debian:/home/sio# apt update
```

```
root@debian:/home/sio# apt upgrade
```

Puis on installe la paquet Squid avec la commande apt install squid 3 :

```
root@debian:/home/sio# apt install squid3
```

Et finalement on installe les paquets complémentaires :

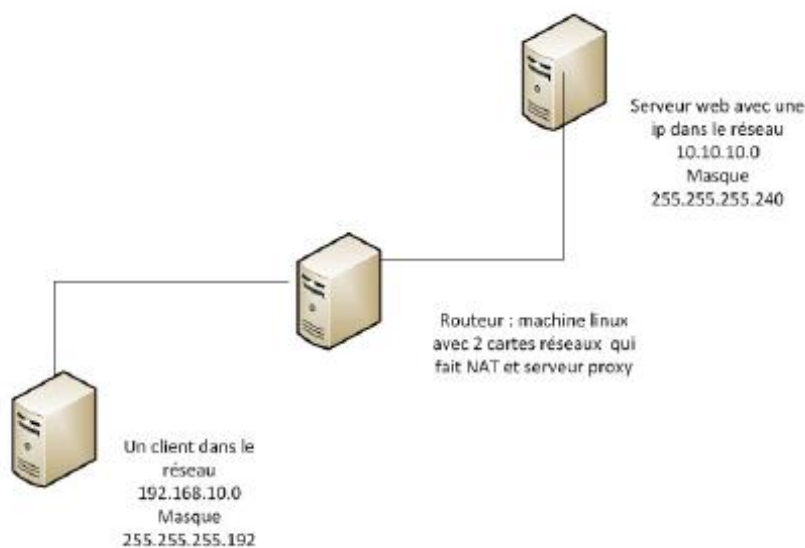
- Squidguard pour la gestion des listes noires avec la commande apt install squidguard :

```
root@debian:/home/sio# apt install squidguard
```

- Et calamaris pour étudier les logs avec la commant apt install calamaris :

```
root@debian:/home/sio# apt install calamaris
```

III- Présentation de l'environnement réseau



Machine	Interface	IP	Masque	Gateway
Serveur Squid	Enp0s3	10.10.10.14	255.255.255.240	
	Enp0s8	192.168.10.62	255.255.255.192	
Serveur Web		10.10.10.4	255.255.255.240	10.10.10.14
Client		192.168.10.2	255.255.255.192	192.168.10.62

Le serveur proxy dispose de deux cartes réseaux, une avec l'adresse 10.10.10.14 /28 et une deuxième avec l'adresse 192.168.10.62 /26. Le serveur Web est sur le réseau 10.10.10.X /28 et dispose de l'adresse IP 10.10.10.4 /28. Le client est sur le réseau 192.168.10.X /26 et dispose de l'adresse IP 192.168.10.2 /26.

A ce moment le client arrive à ping uniquement sa passerelle et donc les deux adresses du serveur proxy mais n'arrive pas à ping et à accéder au serveur Web. Et inversement pour le serveur Web n'arrive pas à ping le client.

Notre but est donc de faire communiquer notre client et notre serveur Web, ainsi le client pourra accéder au réseau 10.10.10.X /28 et pourra accéder aux sites Web. On utilisera un serveur LAMP et on mettra en place un DNS pour les sites web, puis on utilisera iptables et squid.

La configuration réseau du serveur proxy. La carte enp0s3 pour le client et la carte enp0s8 pour le serveur Web :

```
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:76:fe:91 brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.62/26 brd 192.168.10.63 scope global enp0s3
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe76:fe91/64 scope link
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:f0:5d:d2 brd ff:ff:ff:ff:ff:ff
    inet 10.10.10.14/28 brd 10.10.10.15 scope global enp0s8
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fef0:5dd2/64 scope link
        valid_lft forever preferred_lft forever
```

Ici, la configuration réseau du client :

```
Carte Ethernet Ethernet :  
  
Suffixe DNS propre à la connexion. . . :  
Adresse IPv6 de liaison locale. . . . : fe80::50cf:2ac8:e3a8:3b18%2  
Adresse IPv4. . . . . : 192.168.10.2  
Masque de sous-réseau. . . . . : 255.255.255.192  
Passerelle par défaut. . . . . : 192.168.10.62
```

Puis, la configuration réseau du serveur Web :

```
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP  
group default qlen 1000  
    link/ether 08:00:27:ce:71:4d brd ff:ff:ff:ff:ff:ff  
    inet 10.10.10.4/28 brd 10.10.10.15 scope global enp0s3  
        valid_lft forever preferred_lft forever  
    inet6 fe80::a00:27ff:fece:714d/64 scope link  
        __valid_lft__ forever preferred_lft forever
```

IV- Mise en place du NAT

Pour regarder le contenu du routeur, on utilise la commande iptables -t nat -L :

```
root@debian:/home/sio# iptables -t nat -L  
Chain PREROUTING (policy ACCEPT)  
target     prot opt source                destination  
  
Chain INPUT (policy ACCEPT)  
target     prot opt source                destination  
  
Chain OUTPUT (policy ACCEPT)  
target     prot opt source                destination  
  
Chain POSTROUTING (policy ACCEPT)  
target     prot opt source                destination
```

3.1 Mise en place du NAT

Nous allons faire en sorte que le routeur accepte tous les paquets avec les commandes :

iptables -I INPUT -j ACCEPT

iptables -I OUTPUT -j ACCEPT

```
root@debian:/home/sio# iptables -I INPUT -j ACCEPT  
root@debian:/home/sio# iptables -I OUTPUT -j ACCEPT
```

Maintenant, nous allons mettre en place le NAT sur l'IP du côté serveur Web :

```
root@debian:/home/sio# echo 1 > /proc/sys/net/ipv4/ip_forward
```

```
root@debian:/home/sio# iptables -t nat -A PREROUTING -i enp0s3 -p tcp --dport 80 -j DNAT --to-destination 10.10.10.4:80
```

```
root@debian:/home/sio# iptables -A FORWARD -i enp0s3 -p tcp --dport 80 -j ACCEPT
```

```
root@debian:/home/sio# iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE
```

Pour lister toutes les règles iptables pour le NAT, on utilise la commande iptables -t nat -L :

```
root@debian:/home/sio# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source                destination
DNAT       tcp  --  anywhere               anywhere         tcp dpt:http to:10.10.10.4:80

Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source                destination
MASQUERADE all  --  anywhere               anywhere
```

Puis pour lister les règles iptables uniquement on utilise la commande iptables -L :

```
root@debian:/home/sio# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination
ACCEPT     all  --  anywhere               anywhere

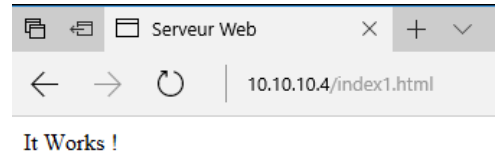
Chain FORWARD (policy ACCEPT)
target     prot opt source                destination
ACCEPT     tcp  --  anywhere               anywhere         tcp dpt:http

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
ACCEPT     all  --  anywhere               anywhere
```

Grâce aux commandes précédentes le client à maintenant accès au serveur Web, il peut pinguer le client et accéder au site, depuis le réseau 192.168.10.X /26 vers le réseau 10.10.10.X /28 :

```
C:\Users\Client-Squid>ping 10.10.10.4

Envoi d'une requête 'Ping' 10.10.10.4 avec 32 octets de données :
Réponse de 10.10.10.4 : octets=32 temps<1ms TTL=63
Réponse de 10.10.10.4 : octets=32 temps=1 ms TTL=63
Réponse de 10.10.10.4 : octets=32 temps=1 ms TTL=63
Réponse de 10.10.10.4 : octets=32 temps=1 ms TTL=63
```



De même, le serveur Web à accès au client :

```
root@TpServeur:/home/sio# ping 192.168.10.2
PING 192.168.10.2 (192.168.10.2) 56(84) bytes of data.
64 bytes from 192.168.10.2: icmp_seq=1 ttl=127 time=0.589 ms
64 bytes from 192.168.10.2: icmp_seq=2 ttl=127 time=1.54 ms
64 bytes from 192.168.10.2: icmp_seq=3 ttl=127 time=1.80 ms
64 bytes from 192.168.10.2: icmp_seq=4 ttl=127 time=1.57 ms
```

3.3 Vérification

Pour vérifier les accès au serveur Web, on peut le lire avec la commande : nano /var/log/apache2/access.log :

(Screen effectué à la fin du TP donc à cette étape le contenu peut être différent)

```

GNU nano 2.7.4          Fichier : access.log
192.168.10.2 - - [08/Apr/2020:15:36:21 +0200] "GET / HTTP/1.1" 200 717 "-" "Moz$
192.168.10.2 - - [08/Apr/2020:15:36:21 +0200] "GET /icons/blank.gif HTTP/1.1" 2$
192.168.10.2 - - [08/Apr/2020:15:36:21 +0200] "GET /favicon.ico HTTP/1.1" 404 4$
192.168.10.2 - - [08/Apr/2020:15:36:21 +0200] "GET /icons/unknown.gif HTTP/1.1"$
192.168.10.2 - - [08/Apr/2020:15:36:21 +0200] "GET /icons/text.gif HTTP/1.1" 20$
::1 - - [08/Apr/2020:15:36:29 +0200] "OPTIONS * HTTP/1.0" 200 126 "-" "Apache/2$
::1 - - [08/Apr/2020:15:36:30 +0200] "OPTIONS * HTTP/1.0" 200 126 "-" "Apache/2$
192.168.10.2 - - [08/Apr/2020:15:36:30 +0200] "GET /index1.html HTTP/1.1" 200 4$
192.168.10.2 - - [08/Apr/2020:15:36:30 +0200] "GET /favicon.ico HTTP/1.1" 404 4$
127.0.0.1 - - [08/Apr/2020:15:46:11 +0200] "GET / HTTP/1.1" 200 728 "-" "Mozill$
127.0.0.1 - - [08/Apr/2020:15:46:12 +0200] "GET /icons/unknown.gif HTTP/1.1" 20$
127.0.0.1 - - [08/Apr/2020:15:46:12 +0200] "GET /icons/text.gif HTTP/1.1" 200 5$
127.0.0.1 - - [08/Apr/2020:15:46:12 +0200] "GET /icons/blank.gif HTTP/1.1" 200 $
127.0.0.1 - - [08/Apr/2020:15:46:12 +0200] "GET /favicon.ico HTTP/1.1" 404 501 $
127.0.0.1 - - [08/Apr/2020:15:46:12 +0200] "GET /favicon.ico HTTP/1.1" 404 501 $
127.0.0.1 - - [08/Apr/2020:15:46:26 +0200] "GET /index1.html HTTP/1.1" 200 434 $
192.168.10.2 - - [08/Apr/2020:15:47:32 +0200] "GET /index1.html HTTP/1.1" 200 4$
192.168.10.2 - - [08/Apr/2020:15:47:33 +0200] "GET /index1.html HTTP/1.1" 200 4$
192.168.10.2 - - [08/Apr/2020:15:47:33 +0200] "GET /index1.html HTTP/1.1" 200 4$

```

V- Refuser l'accès direct au web

Pour refuser les requêtes sur le port 80 on utilise les commandes suivantes qui bloqueront en entrée et sortie l'accès au port 80 (Attention les commandes suivantes vont permettre de ne plus avoir accès au serveur Web depuis le client) :

iptables -A INPUT -p tcp --dport 80 -j DROP

iptables -A OUTPUT -p tcp --dport 80 -j DROP

```

root@TpServeur:/var/www/html# iptables -A INPUT -p tcp --dport 80 -j DROP
root@TpServeur:/var/www/html# iptables -A OUTPUT -p tcp --dport 80 -j DROP

```

VI- Finalisation de l'environnement web

Nous allons mettre en place un DNS sur notre serveur Web Debian, avec deux sites Web. Pour cela on utilisera le paquet Bind9.

On installe le paquet avec la commande Bind 9 avec la commande apt install bind9 :

```

root@TpServeur:/home/sio# apt-get install bind9

```

Modification du hostname : nano /etc/hostname :

```

GNU nano 2.7.4          Fichier : /etc/hostname          Modifié
enzo.serveurwebsquid.tp

```

On ajoute le nouveau du serveur et on associe l'adresse IP de notre serveur DNS à son nom : nano /etc/hosts :

```

GNU nano 2.7.4          Fichier : /etc/hosts          Modifié
127.0.0.1                localhost
127.0.1.1                enzo.serveurwebsquid.tp
10.10.10.4               enzo.serveurwebsquid.tp

```

On indique le domaine et la zone de recherche DNS, ce qui va permettre que notre serveur soit intégré à la zone DNS : nano /etc/resolv.conf

```
GNU nano 2.7.4      Fichier : /etc/resolv.conf      Modifié
# Generated by NetworkManager
search lan
nameserver 10.10.10.4
```

Nous allons désormais nous rendre dans le répertoire /etc/bind qui contient les fichiers de configurations de bind9.

Nous devons déclarer nos zones DNS à savoir la zone « serveurwebsquid.tp » et sa zone inverse associée 10.10.10.in-addr.arpa afin que les adresses IP puissent être traduites en noms de domaines.

On modifie le fichier : nano /etc/bind/named.conf.local

```
GNU nano 2.7.4      Fichier : /etc/bind/named.conf.local      Modifié
//
// Do any local configuration here
//
// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";

zone "serveurwebsquid.tp" {
    type master;
    file "/etc/bind/db.serveurwebsquid.tp";
};

zone "10.10.10.in-addr.arpa" {
    type master;
    file "/etc/bind/db.10.10.10.in-addr.arpa";
};
```

Nous devons maintenant devoir créer nos zones : nano /etc/bind/db.serveurwebsquid.tp

```
GNU nano 2.7.4      Fichier : db.serveurwebsquid.tp      Modifié
$TTL 10800
$ORIGIN serveurwebsquid.tp.
@ IN SOA enzo.serveurwebsquid.tp. root.serveurwebsquid.tp. (
    20160505;
    3h;
    1h;
    1w;
    1h);
@ IN NS enzo.serveurwebsquid.tp.
enzo IN A 10.10.10.4
localhost IN A 127.0.0.1
Enzo IN A 192.168.10.2
```

Nous venons de déclarer notre serveur ainsi que les enregistrements DNS. Nous allons donc configurer les zones DNS inverses : nano /etc/bind/db.10.10.10.in-addr.arpa

```
GNU nano 2.7.4      Fichier : /etc/bind/db.10.10.10.in-addr.arpa      Modifié
$TTL 10800
$ORIGIN 10.10.10.in-addr.arpa.
@ IN SOA enzo.serveurwebsquid.tp. root.serveurwebsquid.tp. (
    20160505;
    3h;
    1h;
    1w;
    1h);
@ IN NS enzo.serveurwebsquid.tp.
100 IN PTR enzo.serveurwebsquid.tp.
```

Pour vérifier les éventuels erreurs sur nos fichiers : named-checkconf -z

```
root@enzo:/home/sio# named-checkconf -z
zone serveurwebsquid.tp/IN: loaded serial 20160505
zone 10.10.10.in-addr.arpa/IN: loaded serial 20160505
zone localhost/IN: loaded serial 2
zone 127.in-addr.arpa/IN: loaded serial 1
zone 0.in-addr.arpa/IN: loaded serial 1
zone 255.in-addr.arpa/IN: loaded serial 1
```

A cette étape la configuration DNS est effectuée et fonctionnelle.

Il ne faut pas oublier de mettre l'adresse IP du serveur Web dans la configuration IP du client dans serveur DNS préféré et depuis le client nous arrivons donc maintenant à ping notre DNS :

☐ Obtenir une adresse IP automatiquement

☒ Utiliser l'adresse IP suivante :

Adresse IP :

Masque de sous-réseau :

Passerelle par défaut :

☐ Obtenir les adresses des serveurs DNS automatiquement

☒ Utiliser l'adresse de serveur DNS suivante :

Serveur DNS préféré :

Serveur DNS auxiliaire :

```
C:\Users\Client-Squid>ping enzo.serveurwebsquid.tp

Envoi d'une requête 'ping' sur enzo.serveurwebsquid.tp [192.168.10.2] avec 32 octets de données :
Réponse de 192.168.10.2 : octets=32 temps<1ms TTL=128
Réponse de 192.168.10.2 : octets=32 temps<1ms TTL=128
Réponse de 192.168.10.2 : octets=32 temps<1ms TTL=128
Réponse de 192.168.10.2 : octets=32 temps<1ms TTL=128
```

Puis faire une résolution DNS depuis le client avec la commande : nslookup

```
C:\Users\Client-Squid>nslookup enzo.serveurwebsquid.tp
Serveur : UnKnown
Address: 10.10.10.4

Nom : enzo.serveurwebsquid.tp
Addresses: 10.10.10.4
           192.168.10.2
```

Configuration du site Web 1 SIO :

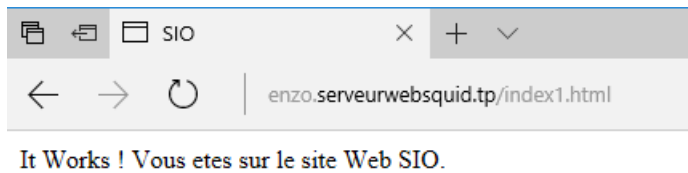
```
GNU nano 2.7.4          Fichier : index1.html          Modifié
<!DOCTYPE html>
<html>
  <head>
    <title>SIO</title>
  </head>
  <body>
It Works ! Vous etes sur le site Web SIO.
  </body>
</html>
```

Configuration du site Web 2 Saint Luc :

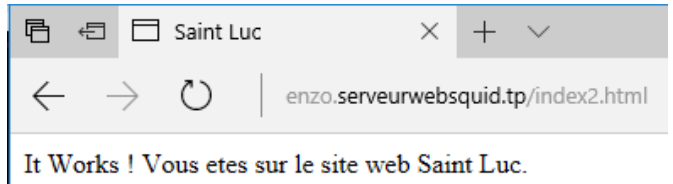
```
GNU nano 2.7.4      Fichier : index2.html      Modifié
<!DOCTYPE html>
<html>
  <head>
    <title>Saint Luc</title>
  </head>
  <body>
    It Works ! Vous etes sur le site web Saint Luc.
  </body>
</html>
```

Finalement, depuis le client la résolution DNS fonctionne bien il peut accéder aux deux différents sites depuis le nom DNS :

Site 1 : SIO



Site 2 : Saint Luc



VII- Installation du serveur Proxy

Pour installer le serveur proxy, on utilise squid.

Installation du paquet squid avec la commande : apt install squid3

```
root@debian:/home/sio# apt install squid3
```

On peut voir que squid utilise deux fichiers de log, nous pouvons les voir : `cd /var/log/squid`

```
root@debian:/var/log/squid# ls
access.log  cache.log  netdb.state
```

Le fichier cache.log recense les actions du service lors de son lancement

Le fichier access.log recense les requêtes traitées par le service :

```
GNU nano 2.7.4 Fichier : cache.log
2020/04/06 15:28:31 kid1| Set Current Directory to /var/spool/squid
2020/04/06 15:28:31 kid1| Starting Squid Cache version 3.5.23 for x86_64-pc-linux-gnu...
2020/04/06 15:28:31 kid1| Service Name: squid
2020/04/06 15:28:31 kid1| Process ID 14524
2020/04/06 15:28:31 kid1| Process Roles: worker
2020/04/06 15:28:31 kid1| With 65535 file descriptors available
2020/04/06 15:28:31 kid1| Initializing IP Cache...
2020/04/06 15:28:31 kid1| DNS Socket created at [::], FD 6
2020/04/06 15:28:31 kid1| DNS Socket created at 0.0.0.0, FD 8
2020/04/06 15:28:31 kid1| Adding domain lan from /etc/resolv.conf
2020/04/06 15:28:31 kid1| Adding domain lan from /etc/resolv.conf
2020/04/06 15:28:31 kid1| Adding nameserver 192.168.1.254 from /etc/resolv.conf
2020/04/06 15:28:31 kid1| Logfile: opening log daemon:/var/log/squid/access.log
2020/04/06 15:28:31 kid1| Logfile Daemon: opening log /var/log/squid/access.log
2020/04/06 15:28:31 kid1| Local cache digest enabled; rebuild/rewrite every 3600/3600 sec
2020/04/06 15:28:31 kid1| Store logging disabled
2020/04/06 15:28:31 kid1| Swap maxSize 0 + 262144 KB, estimated 20164 objects
2020/04/06 15:28:31 kid1| Target number of buckets: 1008
2020/04/06 15:28:31 kid1| Using 8192 Store buckets
2020/04/06 15:28:31 kid1| Max Mem size: 262144 KB
2020/04/06 15:28:31 kid1| Max Swap size: 0 KB
2020/04/06 15:28:31 kid1| Using Least Load store dir selection
2020/04/06 15:28:31 kid1| Set Current Directory to /var/spool/squid
2020/04/06 15:28:31 kid1| Finished loading MIME types and icons.
2020/04/06 15:28:31 kid1| HTCP Disabled.
```

VIII- Mise en place du serveur proxy

On oblige le trafic http à passer par le proxy Squid et nous bloquons dans iptables le trafic http :

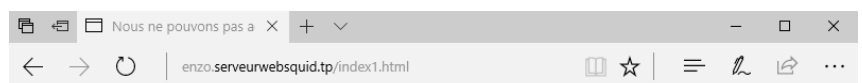
```
root@debian:/home/sio# iptables -t nat -A PREROUTING -i enp0s3 -p tcp --dport 80 -j REDIRECT --to-port 3128
root@debian:/home/sio# iptables -t nat -A PREROUTING -i enp0s3 -s 10.10.10.0/28 -p tcp --dport 80 -j REDIRECT --to-port 3128
root@debian:/home/sio#
```

La première ligne : iptables -t nat -A PREROUTING -i enp0s3 -p tcp --dport 80 -j REDIRECT --to-port 3128 permet d'activer le proxy transparent.

La deuxième ligne : iptables -t nat -A PREROUTING -i enp0s3 -s 10.10.10.0/28 -p tcp --dport 80 -j REDIRECT --to-port 3128 permet d'indiquer à la machine d'intercepter toutes les requêtes sur un port 80, et de les rediriger vers le proxy. Elle est restreint sur le réseau local.

Nous relançons le service squid avec la commande : service squid reload

Maintenant le client n'a plus accès aux sites web :



Désolé, nous ne pouvons pas atteindre cette page

Essayer

- Assurez-vous d'avoir la bonne adresse web : <http://10.10.10.4>
- Rechercher "<http://10.10.10.4>" sur Bing
- Actualiser la page

1.

Pour modifier le port d'écoute sur le port 80, on édite le fichier de configuration de squid : [nano /etc/squid/squid.conf](#)

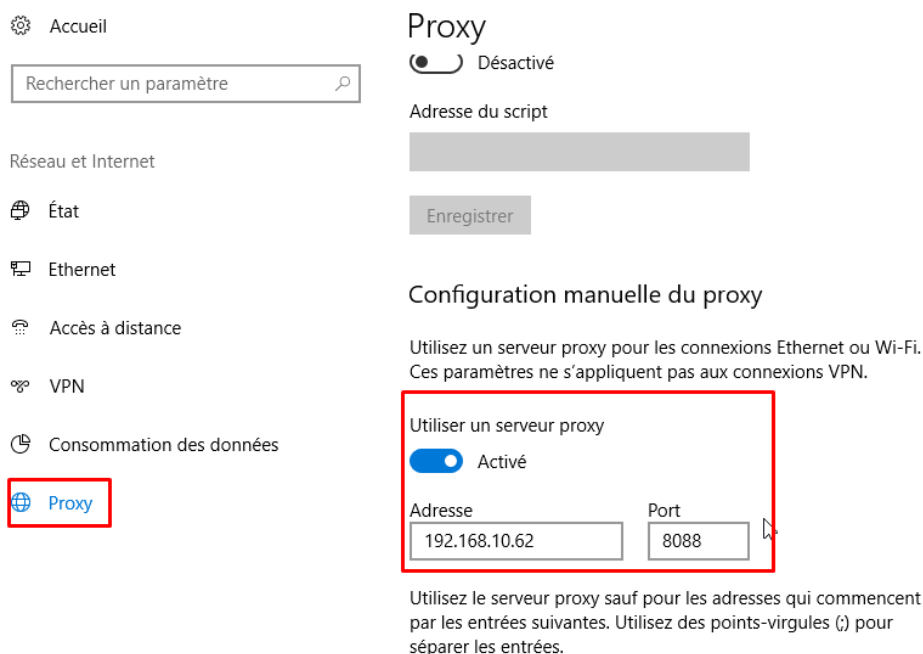
Nous recherchons la ligne du port d'écoute et on change en 8088 :

```
GNU nano 2.7.4      Fichier : /etc/squid/squid.conf      Modifié
#
#       In seconds; idle is the initial time before TCP starts
#       probing the connection, interval how often to probe, and
#       timeout the time before giving up.
#
#       require-proxy-header
#       Require PROXY protocol version 1 or 2 connections.
#       The proxy_protocol_access is required to whitelist
#       downstream proxies which can be trusted.
#
#       If you run Squid on a dual-homed machine with an internal
#       and an external interface we recommend you to specify the
#       internal address:port in http_port. This way Squid will only be
#       visible on the internal address.
#
# Squid normally listens to port 8088_
http_port 8088
```

2.

Pour configurer le client le client avec le proxy, on va dans les paramètres de l'ordinateur > Réseau et Internet > Proxy

On active le proxy et on indique l'adresse IP du serveur ainsi que le port :



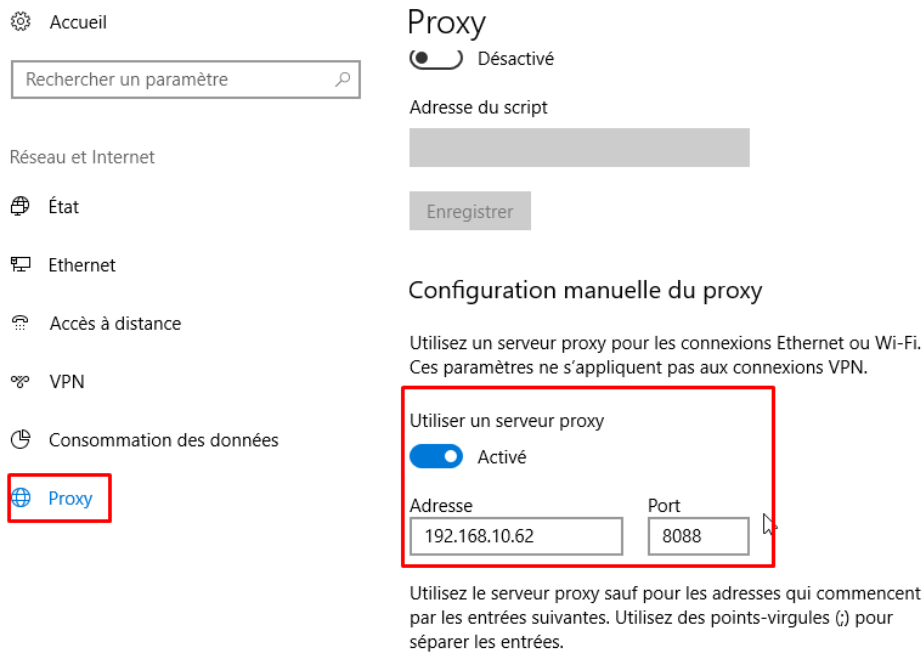
Avec un scan du réseau en utilisant le logiciel Wireshark nous pouvons voir passer le paquet du client transmis vers le serveur Web en passant par le proxy.

3.

Le serveur proxy permet de transmettre les requêtes des clients du réseau 192.168.10.X /26 vers le second réseau 10.10.10.X /28.

4.

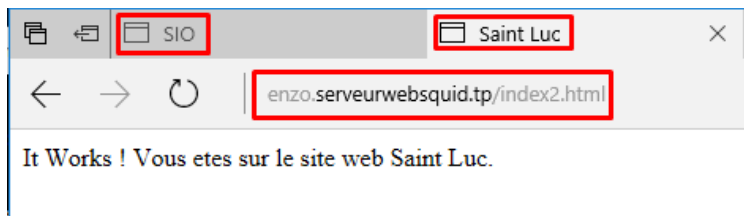
On modifie le client de la même manière qu'effectué précédemment :



5.

Ainsi, notre but est atteint. Il était d'accéder au réseau 10.10.10.X /28 depuis le réseau 192.168.10.X /26 grâce au proxy.

Nous pouvons voir que le client a bien accès aux deux sites Web via le nom DNS et depuis un réseau différent du site Web :



De plus, il arrive à ping l'adresse IP du serveur et le nom DNS :

```
C:\Users\Client-Squid>ping 10.10.10.4

Envoi d'une requête 'Ping' 10.10.10.4 avec 32 octets de données :
Réponse de 10.10.10.4 : octets=32 temps<1ms TTL=63
Réponse de 10.10.10.4 : octets=32 temps=1 ms TTL=63
Réponse de 10.10.10.4 : octets=32 temps=1 ms TTL=63

C:\Users\Client-Squid>ping enzo.serveurwebsquid.tp

Envoi d'une requête 'ping' sur enzo.serveurwebsquid.tp [10.10.10.4] avec 32 octets de données :
Réponse de 10.10.10.4 : octets=32 temps<1ms TTL=63
Réponse de 10.10.10.4 : octets=32 temps=1 ms TTL=63
Réponse de 10.10.10.4 : octets=32 temps<1ms TTL=63
Réponse de 10.10.10.4 : octets=32 temps=1 ms TTL=63
```

De même, le client arrive à ping les deux adresse IP du serveur proxy en plus de l'adresse du serveur Web :

```
C:\Users\Client-Squid>ping 192.168.10.62

Envoi d'une requête 'Ping' 192.168.10.62 avec 32 octets de données :
Réponse de 192.168.10.62 : octets=32 temps<1ms TTL=64
Réponse de 192.168.10.62 : octets=32 temps<1ms TTL=64
Réponse de 192.168.10.62 : octets=32 temps<1ms TTL=64

C:\Users\Client-Squid>ping 10.10.10.14

Envoi d'une requête 'Ping' 10.10.10.14 avec 32 octets de données :
Réponse de 10.10.10.14 : octets=32 temps<1ms TTL=64
Réponse de 10.10.10.14 : octets=32 temps<1ms TTL=64
Réponse de 10.10.10.14 : octets=32 temps<1ms TTL=64
```

Et le serveur Web arrive à ping le client :

```
root@enzo:~# ping 192.168.10.2
PING 192.168.10.2 (192.168.10.2) 56(84) bytes of data.
64 bytes from 192.168.10.2: icmp_seq=1 ttl=127 time=0.590 ms
64 bytes from 192.168.10.2: icmp_seq=2 ttl=127 time=1.56 ms
64 bytes from 192.168.10.2: icmp_seq=3 ttl=127 time=1.25 ms
```

Nous pouvons voir les logs de notre serveur Web avec le fichier access.log présent dans /var/log/apache2 :

```
root@enzo:/var/log/apache2# ls
access.log      access.log.3.gz  error.log.1      error.log.4.gz
access.log.1    access.log.4.gz  error.log.2.gz   other_vhosts_access.log
access.log.2.gz error.log        error.log.3.gz
```

Dans le fichier de log nous pouvons voir les connexions du client à partir de l'autre réseau :

```
GNU nano 2.7.4      Fichier : access.log

192.168.10.2 - - [08/Apr/2020:15:36:21 +0200] "GET / HTTP/1.1" 200 717 "-" "Moz$
192.168.10.2 - - [08/Apr/2020:15:36:21 +0200] "GET /icons/blank.gif HTTP/1.1" 2$
192.168.10.2 - - [08/Apr/2020:15:36:21 +0200] "GET /favicon.ico HTTP/1.1" 404 4$
192.168.10.2 - - [08/Apr/2020:15:36:21 +0200] "GET /icons/unknown.gif HTTP/1.1"$
192.168.10.2 - - [08/Apr/2020:15:36:21 +0200] "GET /icons/text.gif HTTP/1.1" 20$
::1 - - [08/Apr/2020:15:36:29 +0200] "OPTIONS * HTTP/1.0" 200 126 "-" "Apache/2$
::1 - - [08/Apr/2020:15:36:30 +0200] "OPTIONS * HTTP/1.0" 200 126 "-" "Apache/2$
192.168.10.2 - - [08/Apr/2020:15:36:30 +0200] "GET /index1.html HTTP/1.1" 200 4$
192.168.10.2 - - [08/Apr/2020:15:36:30 +0200] "GET /favicon.ico HTTP/1.1" 404 4$
127.0.0.1 - - [08/Apr/2020:15:46:11 +0200] "GET / HTTP/1.1" 200 728 "-" "Mozill$
127.0.0.1 - - [08/Apr/2020:15:46:12 +0200] "GET /icons/unknown.gif HTTP/1.1" 20$
127.0.0.1 - - [08/Apr/2020:15:46:12 +0200] "GET /icons/text.gif HTTP/1.1" 200 5$
127.0.0.1 - - [08/Apr/2020:15:46:12 +0200] "GET /icons/blank.gif HTTP/1.1" 200 $
127.0.0.1 - - [08/Apr/2020:15:46:12 +0200] "GET /favicon.ico HTTP/1.1" 404 501 $
127.0.0.1 - - [08/Apr/2020:15:46:12 +0200] "GET /favicon.ico HTTP/1.1" 404 501 $
127.0.0.1 - - [08/Apr/2020:15:46:26 +0200] "GET /index1.html HTTP/1.1" 200 434 $
192.168.10.2 - - [08/Apr/2020:15:47:32 +0200] "GET /index1.html HTTP/1.1" 200 4$
192.168.10.2 - - [08/Apr/2020:15:47:33 +0200] "GET /index1.html HTTP/1.1" 200 4$
192.168.10.2 - - [08/Apr/2020:15:47:33 +0200] "GET /index1.html HTTP/1.1" 200 4$

[ Lecture de 58 lignes ]
```