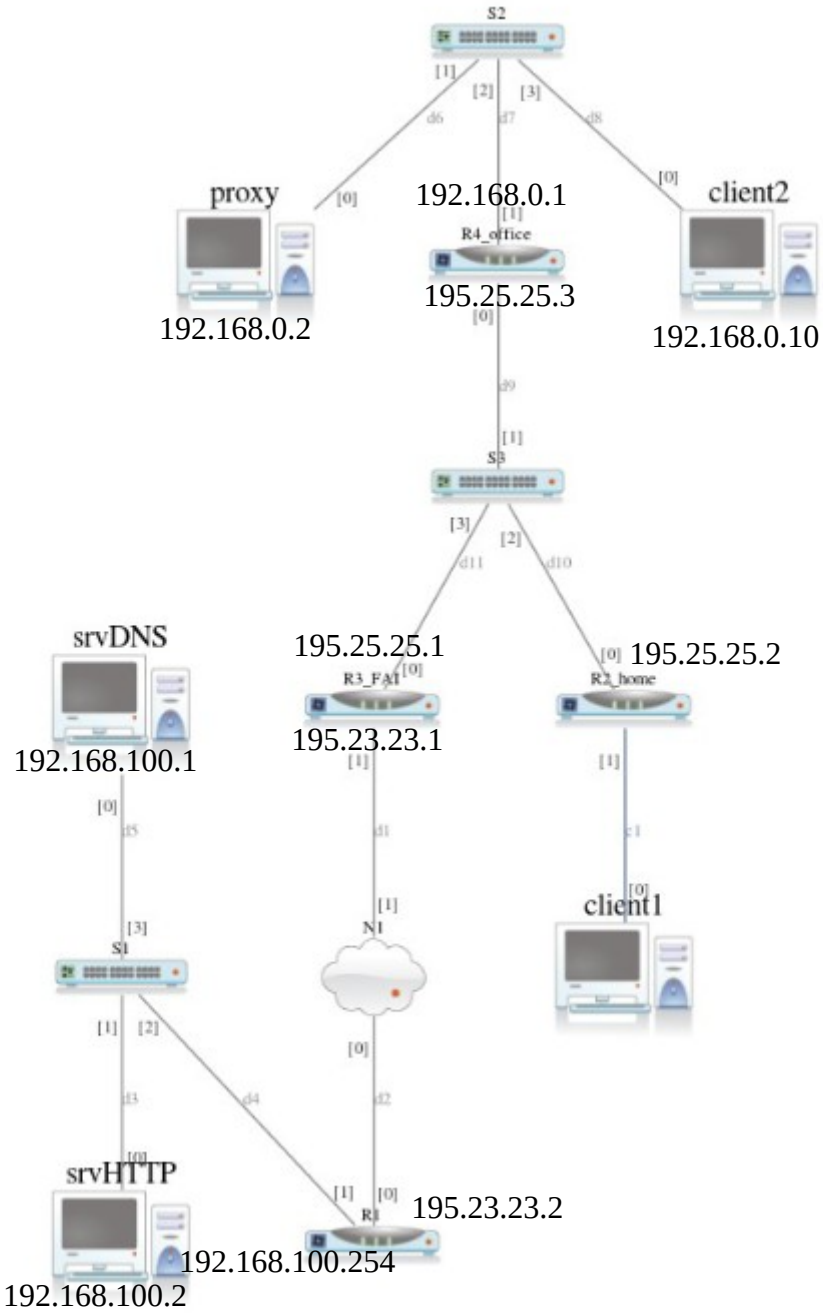
Mathis Ranc
Jean Rouge

CR TP3 NE372

Contexte :
Nous sommes en charge de réaliser l'accès à internet de l'architecture dont le schéma est ci-dessous. Nous pouvons administrer les équipements client1, client2, R1, srvDNS, srvHTTP, R2_home, R4_office et proxy. Le routeur R3_fai est fourni et un fournisseur d'accès et est donc déjà configuré.



NB : Toutes les adresses IP ont un masque de réseau en /24

Mise en œuvre de la solution pour le client1

Le client1 doit pouvoir accèder à internet en utilisant l'adresse IP publique de sa passerelle par défaut. Pour cela on va devoir faire du masquerading pour translater l'adresse IP du client1 en celle du routeur R2_home (195.25.25.2). On doit également configurer la route vers internet.

Pour l'instant le client1 et R2_home n'ont pas d'adresse IP (privée) sur leur réseau commun, on leur en ajoute une manuellement depuis marionnet.

client1 → 10.0.0.1/24
R2_home → 10.0.0.254/24

Ajoutons une route par défaut dans la table de routage du client1 avec ip route :

 # ip route add default via 10.0.0.254

Configurons également la table de routage de R2_home :

 # ip route add default via 195.25.25.1 (route par défaut vers internet)
 # ip route add 192.168.0.0/24 via 195.25.25.3 (@IP R4_office)

Utilisons maintenant nftables pour créer une table dans laquelle on crée une chaîne dans laquelle on ajoute une règle permettant de faire du masquerading :

```
# nft add table nat
# nft 'add chain nat postrouting { type nat hook postrouting priority 100 ; }'
# nft add rule nat postrouting ip saddr 10.0.0.1 oif eth0 masquerade
```

On peut désormais ping l'adresse 193.23.23.2 depuis le client1.
Configurons maintenant le routeur R2_home pour que seuls les protocoles HTTP et DNS puissent sortir :

```
# nft add table ip filter
# nft 'add chain ip filter forward { type filter hook forward priority 0 ; policy drop ; };
# nft insert rule ip forward ct state established counter packets 5 bytes 420 accept
# nft add rule ip filter forward iif eth1 oif eth0 ether type ip tcp dport {53, 80} accept
# nft add rule ip filter forward iif eth1 oif eth0 ether type ip udp dport {53} accept
```

Configuration du serveur DNS

On veut créer la zone titi.fr sur le serveur srvDNS et faire en sorte que cette zone pointe sur l'adresse publique du routeur R1, afin qu'elle soit accessible depuis internet.

Commençons par créer la zone titi.fr, pour cela on configure le fichier /etc/bind/named.conf.local comme ceci :

```
zone "titi.fr" {
    type master;
    file "/etc/bind/db.titi.fr";
};
```

On configure également les RR dans le fichier /etc/bind/db.titi.fr :

```
$TTL 3600
@     IN    SOA     titi.fr.        mail.titi.fr. (
                2007010401          ; Serial
                    3600            ; Refresh [1h]
                     600            ; Retry   [10m]
                   86400            ; Expire  [1d]
                     600 )          ; Negative Cache TTL [1h]
;

@     IN    NS    titi.fr.
titi.fr.      IN    A     193.23.23.2
```

On veut configurer le client1 pour qu'il puisser contacter la zone titi.fr

Problème :
On ne peut pas directement utiliser l'adresse de srvDNS (192.168.100.1) car c'est une adresse privée (séparée de client1 par internet), en pratique on risquerait de ne jamais pouvoir l'atteindre.

On va donc devoir utiliser l'adresse publique de R1 (c'est d'ailleurs pour ça qu'on a fait un enregistrement A qui point sur 192.23.23.2). Pour cela on rajoute la ligne **nameserver 193.23.23.2** dans le fichier /etc/resolv.conf sur le client1.
Cependant cela ne peut pas fonctionner en l'état car R1 n'a aucun moyen de rediriger une requête DNS sur srvDNS.

Mise en œuvre de la solution pour l'accès aux serveurs

Pour pour avoir accès au serveurDNS en connaissant uniquement l'adresse publique de R1, il va falloir configurer la redirection de port sur R1.
On veut que toutes les requêtes entrantes sur le port 53 aillent vers srvDNS, et que toutes les requêtes entrantes sur le port 80 aillent vers srvHTTP.
Pour cela on va utiliser nftables :

```
# nft add table nat
# nft 'add chain nat prerouting { type nat hook prerouting priority -100; }'
```

redirection du port 53 :

```
# nft add rule nat prerouting ip iif eth0 tcp dport 53 dnat to 192.168.100.1
# nft add rule nat prerouting ip iif eth0 udp dport 53 dnat to 192.168.100.1
```

redirection du port 80 :

```
# nft add rule nat prerouting ip iif eth0 tcp dport 80 dnat to 192.168.100.2
```

On modifie le fichier /etc/resolv.conf pour que le serveur de nom par défault soit l'adresse publique de R1.
Et maintenant on peut faire dig sur titi.fr et curl titi.fr pour vérifier que tout fonctionne.

Avec conntrack -L on peut voir la table d'état du NAT et retrouver les différents éléments vu en cours.

Mise en œuvre de la solution pour le client 2

Le trafic internet (http) du client 2 doit passer par un proxy.
On ne veut donc pas faire de translation d'adresse pour ce client.
En revanche on veut modifier sa route par défaut par l'IP du proxy, ainsi tout le trafic à destination extérieure passera par ce proxy.

On effectue donc la translation d'adresse du proxy, de la même manière qu'on l'a déjà fait pour le client1 précédemment.
Concernant les flux à autoriser :
Le client 2 n'a pas besoin de résolveur DNS car c'est le proxy qui se chargera de la résolution des noms. Cependant on doit quand même autoriser le flux DNS dans le routeur car le proxy en a besoin lui.

Le proxy HTTP utilise le package tinyproxy. Par défaut il écoute sur le port 8888.
Celui-ci est configurable dans le fichier /etc/tinyproxy/tinyproxy.conf .

Pour que wget utilise bien le proxy comme on le souhaite, il suffit de mettre l'IP du proxy et son port dans la variable d'environnement http_proxy.
export http_proxy="ip_proxy[:port]**"**

L'avantage de l'utilisation d'un proxy est avant tout l'ajout d'un niveau de sécurité, en effet les machines du réseau local ne sont maintenant plus directement exposé à internet. De plus le proxy est plus facile à mettre en place, la configuration sur les postes utilisateurs est minimale.
Les inconvénients seraient la limitation des protocoles disponible, ici notre proxy peut uniquement faire passer des requêtes/réponse HTTP. Et il est possible que les performances du réseau se retrouvent affectée, si par exemple beaucoup de trafic passe par un unique proxy et qu'il fasse du filtrage sur chacun des paquets, cela peut vite devenir non-négligeable.