Déploiement de l'IPv6 à la Maisel

Équipe :
Maël Kimmerlin
Andrei Mihaescu
Alexis Mousset
Anthony Vérez

Encadrants :
Jehan Procaccia
Aurélien Guerson

www.minet.net

Ce document décrit le déploiement de l'IPv6 chez le FAI associatif MiNET, recevant du routing et du traffic d'un AS dont il dépend. Nous fournissons un accès et des services internet sur le campus de Télécom SudParis et Télécom Ecole de Management. Notre volonté a été d'effectuer un déploiement dual stack affectant le moins possible le bon fonctionnement de notre réseau et ayant une sécurité comparable à l'IPv4 étant donné que ce déploiement concerne la totalité de nos adhérents. Nous partageons dans ce document de nombreux retours d'expérience.

Remerciements

Nous tenons à remercier Jehan Procaccia et Aurélien Guerson, administrateurs réseaux de notre DSI, pour toute l'aide qu'ils nous ont apporté au cours de ce projet ainsi que Eric Collery, directeur de la DSI.

Nous tenons aussi à remercier Clément Bethuys, Florian Duraffourg et Mehdi Sebbar, vétérans de l'association MiNET, pour leurs conseils précieux et leur grand intérêt pour le projet.

Table des matières

1. Introduction

1.1 But du projet

Ce projet s'est déroulé au sein de l'association MiNET, fournisseur d'accès internet associatif du campus des écoles Télécom SudParis et Télécom Ecole de management. L'objectif du projet était de déployer l'IPv6 sur l'ensemble du réseau pour fournir les mêmes services qu'en IPv4, aussi bien en attribuant une IPv6 publique aux adhérents (abonnés) qu'en rendant nos services publics accessibles en IPv6. Une attention particulière a été portée à la sécurité, pour atteindre le même niveau qu'en IPv4.

1.2 Présentation

1.2.1 L'association MiNET

MiNET (Maisel INT Network) est une association à but non lucratif fournissant un accès internet à l'ensemble des étudiants logés sur le campus de Télécom SudParis et Télécom Ecole de Management. MiNET propose une connexion filaire et Wi-Fi ainsi que plusieurs services comme de l'hébergement web, de la téléphonie sur IP, la télévision sur le réseau, etc.

1.2.2 L'équipe

L'équipe du projet est composée de :
- Maël Kimmerlin
- Andrei Mihaescu
- Alexis Mousset
- Anthony Vérez

1.2.3 Les tuteurs

Les tuteurs nous ayant aidé sur ce projet sont :
- Aurélien Guerson
- Jehan Procaccia

2. Présentation du réseau et des services en IPv4

2.1 L'environnement matériel et logiciel

2.1.1 Les équipements réseaux

Notre cœur de réseau est un switch cisco catalyst c4500 utilisant une image entservices 15.1(2). Chaque bâtiment du campus a deux ou trois switchs cisco catalyst c2960G en cascade avec des images lanbase 15.0(1) SE2 . Nous avons voulu mettre à jour l'ensemble de notre parc de switchs 2960 vers la version 15.0 (2) pour pouvoir appliquer les access-list mais nous avons eu des problèmes de mémoire sur les rcom0 de tous les bâtiments avec la version SE2. Suite à la sortie de la version SE3, nous envisageons la mise à jour si les tests sont probants. Nous avons aussi dans un bâtiment un switch cisco catalyst c3750G-12S avec une image ipservice 12.2(55) SE5 et des switchs c3750G-48PS stackés avec des images ipbase 15.0(1) SE2. Pour notre salle serveur nous avons aussi un switch cisco catalyst c3750G-48PS.

2.1.2 La virtualisation

À part les serveurs diffusant la télévision, le parefeu et l'IDS, tous les serveurs sont virtualisés. Nous utilisons la solution de virtualisation Proxmox VE[**?**] (sous licence GNU GPL), qui permet de gérer des containers OpenVZ et des machines virtuelles KVM. Au niveau du réseau, les machines sont configurées en bridge. Ainsi la configuration des interfaces réseau se fait dans le machine et non pas sur l'hôte.

2.2 Le plan d'adressage

Voici le plan d'adressage des IP publiques du réseau MiNET.
Nous disposons des plages 157.159.40.0/21 et 2001 :660 :3203 :400 : :/57.

vlan	IPv4	IPv6	Commentaires
2	157.159.40.0/25	2001 :660 :3203 :422 : :/64	Serveurs de Production
3	157.159.40.128/26	2001 :660 :3203 :423 : :/64	Serveurs de Développement
5	157.159.40.224/28	2001 :660 :3203 :425 : :/64	Sous-réseau Sadint
27	192.168.27.0/24	2001 :660 :3203 :427 : :/64	Serveurs de diffusion TV
41	157.159.41.0/24	2001 :660 :3203 :401 : :/64	Filaire Bâtiment U1
42	157.159.42.0/24	2001 :660 :3203 :402 : :/64	Filaire Bâtiment U2
43	157.159.43.0/24	2001 :660 :3203 :403 : :/64	Filaire Bâtiment U3
44	157.159.44.0/24	2001 :660 :3203 :404 : :/64	Filaire Bâtiment U4
45	157.159.45.0/24	2001 :660 :3203 :405 : :/64	Filaire Bâtiment U5
46	157.159.46.0/24	2001 :660 :3203 :406 : :/64	Filaire Bâtiment U6
47	157.159.47.0/24	2001 :660 :3203 :40F : :/64	Filaire Foyer
400	157.159.40.252/30	2001 :660 :3203 :400 : :/64	Interconnexion DSI
426	157.159.40.240/29	2001 :660 :3203 :426 : :/64	Dépôts VLC
430	10.6.30.0/24	2001 :660 :3203 :430 : :/64	Tests Ipv6

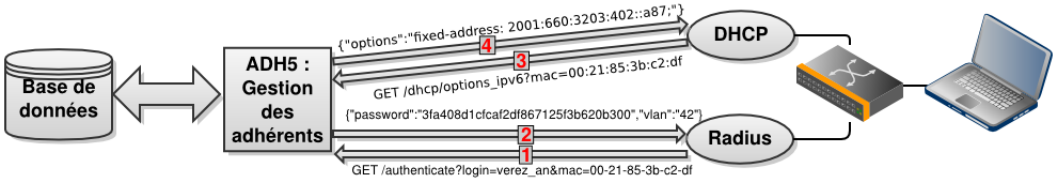
Nous avons modifié le plan d'adressage en IPv6 par rapport à celui en IPv4 pour des raisons de fonctionnement interne. Ayant plus d'IP en IPv6, nous avons voulu améliorer la hiérarchie de notre adressage.

2.3 Gestion des adhérents et authentification

Nous utilisons un logiciel web développé en interne pour gérer l'ensemble des adhérents, de leurs machines et de leurs cotisations (il faut adhérer pour avoir accès à internet par nos soins). Cela nous permet d'utiliser une unique base de données pour l'authentification et l'attribution d'adresses IP.

Pour l'authentification des adhérents, nous utilisons un serveur RADIUS (utilisant le standard 802.1x). Celui-ci a été modifié avec un patch pour être interfacé avec notre logiciel de gestion des adhérents. A chaque fois qu'un adhérent se connecte, une requête est envoyée au RADIUS qui envoie une requête en HTTP à notre logiciel qui vérifie alors dans sa base de données si l'adhérent est à jour de cotisation, son login existe, son mot de passe est correct et si l'adresse MAC de l'ordinateur essayant de s'authentifier est présente dans la liste des adresses MAC de l'adhérent. Le logiciel de gestion des adhérents (nommé ADH5) envoie alors sa réponse au format JSON au serveur RADIUS qui permet ou non la connexion.

Une fois l'authentification RADIUS effectuée, l'adhérent peut demander une IP au serveur DHCP. Nous souhaitons que chaque adhérent ait une adresse IP fixée, servie par DHCP car nous voulons être capables de parcourir rapidement l'historique d'attribution des adresses IP afin de pouvoir plus facilement identifier un adhérent en cas de problème. Pour cela, nous attribuons des adresses IP statiques à nos adhérents. A cet effet, nous avons donc ajouté un patch au serveur ISC [1] DHCP permettant de faire le même genre de requête HTTP pour obtenir l'IP à attribuer à l'adhérent.



2.4 Autres services

Nous proposons d'autres services aux adhérents comme un hébergement de sites web, des listes de diffusion, de la téléphonie sur IP ou la diffusion de la télévision en multicast. Ce dernier service se base sur la réception et la rediffusion de la TNT.

Outre les services aux adhérents, nous avons de nombreux services internes à l'association : l'ensemble des services mails (imap, smtp, mx), un logiciel de gestion du réseau, un wiki, etc.

1. Internet Systems Consortium, http://www.isc.org

3. Stratégie de migration vers l'IPv6

3.1 Concernant les adhérents

Nous avons décidé de faire des vlans en dual stack, c'est-à-dire IPv4 et IPv6, pour les adhérents. Nous leur attribuons des adresses publiques en IPv4 et en IPv6 pour les connexions filaires. Nous n'avons pas changé notre mécanisme d'authentification, nous avons ajouté un service d'attribution d'adresse IPv6.

Un problème rapidement soulevé a été celui de l'autoconfiguration utilisant les Router Advertisement. Les Router Solicitation (RS) sont des requêtes réalisées par un hôte afin de configurer l'IPv6. Le serveur de Router Advertisement (RA) répond un RA contenant le préfixe du réseau ainsi que le type d'attribution des IPs : par autoconfiguration basée sur l'adresse MAC ou par DHCP. Nous avons commencé par bloquer tous les RA au niveau du routeur mais ceci empêchait la configuration automatique des hôtes. Nous avons donc décidé de modifier les RA du routeur pour que les machines des adhérents fassent des requêtes DHCP et pas d'autoconfiguration. Nous bloquons tous les autres RA.

3.2 Les services de l'association

En ce qui concerne les sites web de l'association, on y accède à travers un reverse proxy. C'est-à-dire que toutes les requêtes arrivent sur une machine qui les transfère au serveur web concerné selon le site web demandé. Nous avons attribué une adresse IPv6 uniquement au reverse proxy qui transfère ensuite en IPv4 aux serveurs web. L'ensemble de nos sites web ont donc rapidement été configurés pour être accessibles en IPv6.

Le Wi-Fi étant en pleine restructuration, nous avons décidé de ne pas faire la migration en IPv6. Ce sera sans doute un prochain défi l'association.

Pour la télévision sur IP, nous avons fait la migration en IPv6 et nous avons constaté une augmentation du débit nécessaire. Nous devons en effet garder la télévision en IPv4 pour nos adhérents qui ne disposeraient pas d'une connexion en IPv6, à cause d'un système d'exploitation que ne dispose pas de cette technologie par exemple (avant Windows XP inclus chez le monde Microsoft). Tous les adhérents ayant une adresse IPv4, nous avons décidé de ne diffuser qu'en pour le moment IPv4, même si nous sommes prêt à diffuser en IPv6, pour que tout le monde puisse recevoir la télé sans doubler le trafic.

3.3 Les vlans d'administration

Nous avons décidé de ne pas passer nos vlans d'administration en IPv6 puisque nous n'en avons pas encore besoin et que nous savons ne pas pouvoir apporter la même sécurité pour l'instant en IPv6 qu'en IPv4 pour cette problématique. Toutes nos interfaces d'administration (pour gérer nos serveurs et équipements réseau) resteront en IPv4, dans un réseau privé accessible à distance derrière un VPN.

4. Mise en place de l'IPv6

4.1 Tests et mise en place de l'infrastructure réseau

Nous avons commencé par passer tous nos switchs en mode dual stack pour avoir les jeux de commandes en IPv6. Nous avons réalisé des mises à jour des images IOS de nos switchs Cisco afin d'obtenir plus de fonctionnalités IPv6. Dans l'ensemble les équipements semblent matures sauf concernant ...

Nous avons ensuite mis en place le routage avec notre DSI. Les protocoles de routage dynamique ont tous une version spécifique à l'IPv6. Nous utilisons une route statique pour l'upload pour des raisons de pare-feu et du RIPng pour le download. Voici la configuration de notre routeur :

```
ipv6 unicast-routing
ipv6 multicast-routing
!//pour les RA, NA, ND etc
ipv6 multicast rpf use-bgp
!
interface Vlan400
 ipv6 address 2001:660:3203:400::1/64
 ipv6 rip RIP_MINET enable
!//Ripng pour le download, annonces vers la DSI
!
ipv6 route ::/0 Vlan400 2001:660:3203:400::2
!//Pour l'upload
ipv6 router rip RIP_MINET
 distribute-list prefix-list minet-to-disi out Vlan400
!//Configuration du RIP
!
ipv6 prefix-list minet-to-disi seq 80 permit 2001:660:3203:430::/64
!//Definition des reseaux a annoncer
```

Nous avons alors fait les premiers tests en IPv6 seulement dans le vlan 430 (tests IPv6). Le vlan 400 est un vlan d'interconnexion avec notre DSI.

4.1.1 Configuration des switchs

Nous avons activé le dual stack sur l'ensemble de nos switchs

```
switch-u1(config)#sdm prefer dual-ipv4-and-ipv6 default
switch-u1(config)#do show sdm prefer
 The current template is "dual-ipv4-and-ipv6␣default" template.
 The selected template optimizes the resources in
 the switch to support this level of features for
 0 routed interfaces and 255 VLANs.
```

4.1.2 Gestion des Router Advertisement sur Cisco IOS

Souhaitant supprimer les Router Advertisement indésirables, nous avons mis en place des RAGUARD policy, autrement dit, des politiques de contrôles des RA transitant par le routeur, un Cisco 4503.

```
 ipv6 nd raguard policy RAGUARD
  device-role switch
  managed-config-flag on
  other-config-flag on
  router-preference maximum low
```

La commande device-role donne le rôle de l'équipement de l'autre côté du lien du port sur lequel est appliquée la politique. Le "managed-config-flag on" active le flag managed-config dans le RA envoyé, les router advertisement dont le flag n'est pas présent est supprimé, de même pour le flag other-config. La commande "router-preference maximum low" indique que le niveau de préférence indiqué dans le router advertisement ne doit pas être supérieur au niveau low. Les deux dernières lignes de configuration sont facultatives.

Seul notre routeur gère les RAGuard, les autres équipements (Cisco 3750 et Cisco 2950) n'implémentant pas cette fonctionnalité. Nous avons donc dû procéder avec des access-lists :

```
ipv6 access-list ra-block
deny icmp any any  router-advertisement
permit ipv6 any any
```

Cette access-list a été appliquée sur tous les ports de tous les switchs sauf les liens de uplink vers le routeur.

Nous avons ainsi supprimé l'ensemble des RA indésirables. Nous avons alors mis en place nos propres RAs dans les vlans destinés à passer en IPv6.

4.1.3 Configuration des machines virtuelles

Pour notre cluster, nous utilisons Proxmox VE avec deux types de virtualisation, OpenVZ ou KVM. Au niveau réseau, nos machines virtuelles sont configurées en bridge. Ainsi, il n'a pas été nécessaire d'attribuer une IPv6 aux machines hôtes. Les VLAN publics de serveurs (dans 157.159.40.0/24 en IPv4) sont tous en dual stack.

Extrait d'un /etc/network/interfaces typique, **eth2** étant l'interface dans le VLAN 2, configurée par Proxmox VE (fichier de configuration réseau spécifique à Debian) :

```
auto eth2

iface eth2 inet static
        address 157.159.40.18
        netmask 255.255.255.128
        gateway 157.159.40.1

iface eth2 inet6 static
        address 2001:660:3203:422::a18
        netmask 64
        gateway 2001:660:3203:422::1
```

On peut remarquer la convention suivie pour le choix des IPv6 : Afin de mieux faire la correspondance pour entre les IPv4 et IPv6 de nos machines nous avons choisi d'identifier en IPv6 les hôtes par *: :a* suivit de l'identifiant hôte IPv4. Le fait de ne pas utiliser l'autoconfiguration permet de renforcer la protection de la vie privée de nos adhérents sur internet. En effet, cette adresse serait basée sur l'adresse MAC de leur carte réseau ce qui permettrait de les identifier même si ils se connectent depuis deux réseau différents. Il existe des extensions à l'autoconfiguration pour empêcher ce pistage mais elles ne nous ont pas semblé assez matures.

Cependant pour les machines virtuelles KVM, nous avons eu quelques soucis de routes par défaut. Aucune route ne se paramètre quand on donne l'adresse IP pour l'IPv6. Nous les avons résolus en lançant systématiquement au démarrage cette commande :

```
ip -6 route add default via 2001:660:3203:422::1 dev eth2
```

4.1.4 Tests

Nous avons réalisé quelques tests initiaux avec des adresses IP statiques dans un vlan dédié et nous avons dressé une petite liste des sites accessibles en IPv6. Les sites de Google et Facebook sont disponibles en IPv6 et représentent la très grande majorité du trafic IPv6 pour l'instant. Les miroirs de mise à jours des distributions Linux sont aussi maintenant pratiquement tous accessibles en IPv6, tout comme les acteurs majeurs en matière de mail (Gmail, Microsoft, Yahoo).

4.2 Migration des serveurs

4.2.1 Serveurs DNS

Nous utilisons le logiciel libre Bind9 pour gérer nos DNS.

On ajoute dans **/etc/bind/named.conf.options** :

```
listen-on-v6 {
            2001:660:3203:422::A54;
};
```

Après la directive listen pour l'IPv4. Et on ajoute les IPv6 de notre DSI (les nôtres étant incluses dans ce réseau) :

```
acl int {
        157.159.0.0/16;
        2001:660:3203::/48;
};
```

Pour ajouter une entrée, dans un des fichiers des IP publiques :

```
smtp                    IN AAAA 2001:660:3203:422::A18
```

Pour le reverse, dans **/etc/bind/named.conf** sur le master :

```
zone "2.2.4.0.3.0.2.3.0.6.6.0.1.0.0.2.ip6.arpa" {
        type master;
        file "/etc/bind/zones/minet/rev/minet.net.rev.public-vlan-ipv6";
};
```

Puis dans **/etc/bind/zones/minet/rev/minet.net.rev.public-vlan-ipv6** :

```
;
; 2001:660:3203:422::/64
;
; Zone file built with the IPv6 Reverse DNS zone builder
; http://rdns6.com/
;
$TTL 1h ; Default TTL
@       IN      SOA     ns1.minet.net. dns.minet.net. (
        2013052201      ; serial
        21600           ; slave refresh interval
        3600            ; slave retry interval
        3600000         ; slave copy expire time
        600             ; NXDOMAIN cache time
        )

                IN      NS      ns1.minet.net.
                IN      NS      ns2.minet.net.
                IN      NS      ns0.xname.org.
                IN      NS      ns1.xname.org.
                IN      NS      ns2.xname.org.
                IN      NS      diamant.int-evry.fr.

;
; domain name servers
;
;@      IN      NS      ns1.minet.net.


; IPv6 PTR entries

$ORIGIN 2.2.4.0.3.0.2.3.0.6.6.0.1.0.0.2.ip6.arpa.

8.1.a.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0    IN    PTR    smtp.minet.net.
5.2.a.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0    IN    PTR    mx1.minet.net.
6.2.a.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0    IN    PTR    mx2.minet.net.
5.5.a.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0    IN    PTR    ns1.minet.net.
4.5.a.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0    IN    PTR    ns2.minet.net.
3.0.1.a.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0    IN    PTR    revproxy.minet.net.
2.7.a.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0    IN    PTR    imap.minet.net.
9.9.a.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0    IN    PTR    proxy-ipv6.minet.net.
```

Nous pouvons vérifier la syntaxe avec :

```
$ named-checkzone 2.2.4.0.3.0.2.3.0.6.6.0.1.0.0.2.ip6.arpa /etc/bind/zones/
   minet/rev/minet.net.rev.public-vlan-ipv6
$ named-checkzone minet.net. /etc/bind/zones/minet/db/minet.net.db.ALL
```

4.2.2 Serveurs web

Nous utilisons un reverse proxy (Nginx), nous n'avons eu qu'à configurer celui-ci puisqu'il sert l'ensemble des sites web de MiNET. Nous avons donc, après avoir vérifié que la version de nginx installée supporte l'IPv6, ajouté la directive suivante dans tous les fichiers de configuration présents dans /etc/nginx/sites-available (en adaptant pour https) :

```
listen                  [::]:80 default;
```

Oui, ce fut aussi simple que ça.

4.2.3 Serveurs mail

Postfix

Le MTA installé sur tous les serveurs mail est *Postfix*. Il gère l'IPv6[3] depuis la version 2.2 datant de 2005, et l'active par défaut depuis la 2.9, qui est installée sur nos serveurs d'échange, les seuls à avoir des IP publiques. Pour spécifier une valeur différente, il faut utiliser dans **/etc/postfix/main.cf** le paramètre :

```
inet_protocols = ipv4
```

Toutes les configurations spécifiques concernent des serveurs et des adresses internes restées en IPv4, et ne nécessitaient pas de modification. Nous avons dû nous limiter à l'IPv4 attendant d'obtenir la délégation du reverse-DNS en IPv6, car l'absence d'entrée DNS inverse peut faire suspecter un serveur spammeur.

Il est aussi intéressant de noter que l'IPv6 pose de nouveaux problèmes concernant le spam[2]. Il est existe aussi un draft de RFC[1] concernant la transition des systèmes mails des FAI vers l'IPv6 décrivant les différentes étapes de la migration et les problèmes les plus souvent rencontrés.

Cyrus

Cyrus est le serveur IMAP/Sieve utilisé à MiNET. Il a suffit de dupliquer les lignes de configuration des adresses en IPv4 pour l'IPv6 :

```
SERVICES {
  ...
  imap      cmd="imapd␣-U␣30" listen="[157.159.40.27]:imap" prefork=1 maxchild
     =120
  imap6     cmd="imapd␣-U␣30" proto=tcp6 listen="[2001:660:3203:422::a27]:imap
     " prefork=1 maxchild=120
  ...
  sieve     cmd="timsieved" listen="[157.159.40.27]:sieve" prefork=0 maxchild
     =100
  sieve6    cmd="timsieved" proto=tcp6 listen="[2001:660:3203:422::a27]:sieve"
      prefork=0 maxchild=100
  ...
}
```

4.2.4 Pare-feu

Vous trouverez en annexe A le script iptables que nous avons réalisé. Pour activer ce script au démarrage de la machine, il faut entrer la commande suivante en tant qu'utilisateur root :

```
update-rc.d firewallv6 defaults
```

Nous avons dû porter une attention particulière au filtrage de l'ICMPv6 qui permet bien sûr de faire fonctionner le ping mais aussi le RA, RS, ND, NA, multicast, etc. Nous avons par exemple régulièrement des rogues RA venant de l'extérieur qui sont bloqués sur le pare-feu.

Au niveau du filtrage, nous bloquons par défaut tous les ports en dessous du port 2000 car ils sont connus pour être utilisés par des services réseau. Il y a un filtrage des IP qui entrent et qui sortent, on bloque les paquets avec des IP manifestement fausses. Des ports liés à des logiciels de P2P ou à des vers bien connus sont bloqués. Bien sûr, nous nous protégeons du scan de ports. Le filtrage a été testé avec les logiciels nmap et scapy.

Le trafic entrant est copié sur une autre machine liée directement (sans passer par un switch) avec la fonctionnalité TEE de iptables destinée justement à copier les paquets arrivant sur le pare-feu vers une autre adresse IP.

Il est à noter que nous ne pouvons pas utiliser la fonctionnalité rpfilter (Reverse Path Filtering) car nous n'avons pas un noyau >= 3.3

4.2.5 Ndpmon

Pour offrir plus de sécurité sur notre réseau nous avons décidé de mettre en place un outil de sécurité qui sert à détecter les RA envoyés par une source non autorisée, un adhérent dans son logement par exemple. NDP correspond en IPv6 à ARP en IPv4. De nombreuses attaques sont possibles sur le protocole ARP, c'est davantage encore le cas en IPv6 avec NDP. L'outil que nous avons mis en place est le logiciel libre Ndpmon qui peut être vu comme l'équivalent en IPv6 de ARPwatch (IPv4).

Installation

Pour l'installer il faut télécharger les sources à partir de site http://ndpmon.sourceforge.net, les copier sur la machine sur laquelle vous voulez le déployer et utilisez le script *./configure* avec les options pour spécifier le dossier de l'installation et du fichier de configuration. Il dispose aussi d'une interface web qui facilite la surveillance de son activité. Après il suffit juste de faire un *make* et *make install* pour l'installer.

Configuration

Son fonctionnement est simple. Il peut être configuré de deux manières :
- Statique
- Dynamique

Configuration statique Après l'avoir installé, dans le fichier de configuration il faut définir toutes les sources qui seront autorisées a émettre des paquets ICMPv6. Pour plus de détails vous pouvez consulter la documentation officielle sur le site http ://ndpmond.sourceforge.net. Cependant, nous avons choisi d'utiliser le mode dynamique de configuration.

Configuration dynamique La configuration dynamique a l'avantage de se faire automatiquement, mais par contre elle doit être faite à une période pendant laquelle vous êtes sûr qu'aucune attaque n'est pas en train d'être réalisée. Une fois cette condition assurée vous pouvez spécifier dans le fichier de configuration sur quelle interface il doit écouter. Après il suffit juste de tapez la commande *ndpmon -L* et vous allez le démarrer en *Learning mode*, c'est à dire en apprentissage. Vous attendez quelque minutes pour qu'il découvre toutes les sources. Ensuite, vous le lancez en mode normal *ndpmon* et il va commencer à écouter sur les interfaces que vous avez spécifiées.

4.2.6 Proxy

Nous avons mis en place un serveur proxy squid3. A partir de la version 3.1 l'IPv6 est activé par défaut. Nous avons effectué des tests en configurant le serveur manuellement sur nos ordinateurs et après nous avons regardé les logs pour voir que tout se passe bien. Le but de ce serveur est de diminuer le trafic web et également d'offrir à nos adhérents des réponses plus rapides pour les pages web très visitées.

Pour configurer le proxy, nous avons spécifié le port sur lequel il écoute. En respectant les règles spécifiques à l'association MiNET le port a été modifié à 81.

```
http_port 81 tproxy
```

Nous avons inclus des règles permettant l'accès au serveur qu'aux hôtes autorisés :

```
acl localnet src 2001:660:3203:423::/64 2001:660:3203:430::/64
```

Après avoir bien testé le fonctionnement du serveur, l'étape suivante a été de le rendre transparent, parce qu'il serait plus commode pour les adhérents de ne rien configurer de plus. Nous n'avons pas réussi rendre le proxy transparent, cependant nous allons vous expliquer comment ça aurait dû fonctionner. Nous avons essayé d'utiliser la fonctionalité TPROXY de squid et de iptables. Celui-là permet l'interception des paquets sans modifiant l'entête IP (comme par exemple le NAT/PAT, qui altère l'information de niveau 3 et 4) et donc une meilleure transparence.

En suivant les conseils de la documentation officielle de squid http ://wiki.squid-cache.org/Features/Tproxy4 nous avons configuré sur cette machine aussi un ip6tables pour changer le port 80 en port 81, le numéro du port sur lequel notre serveur proxy écoute et pour éviter que la même requête passe deux fois par le proxy.

```
ip6tables -t mangle -N DIVERT
ip6tables -t mangle -A DIVERT -j MARK --set-mark 1
ip6tables -t mangle -A DIVERT -j ACCEPT
ip6tables  -t mangle -A PREROUTING -p tcp -m socket -j DIVERT
ip6tables  -t mangle -A PREROUTING -p tcp --dport 80 -j TPROXY --tproxy-mark 0
   x1/0x1 --on-port 81
```

Étant donné notre expérience de squid en IPv4, nous avons cherché sur la documentation Cisco et trouvé que le protocole WCCP (Web Cache Communication Protocol), qui permet l'intégration des serveurs de cache web, en version 2 supporte aussi l'IPv6. La configuration d'un tel service est très directe et simple.

Il faut d'abord activer le service WCCP en utilisant la commande :

```
  Router(config)# ipv6 wccp web-cache
```

pour activer le service. Après il suffit juste d'aller sur chaque interface et de spécifier que le trafic qui arrive sur l'interface sera redirigé vers un serveur de cache :

```
 Router(config)# interface ethernet 0/1
  Router(config-if)# ipv6 wccp web-cache redirect in
```

Notre équipement, servant de passerelle, est un switch Catalyst 4503 qui supporte pas les commandes pour configurer ce service. Nous avons essayé une autre solution utilisant le "policy routing" :

Nous avons configuré un access-list pour filtrer deux hôtes de test, simulant des ordinateurs d'adhérents :

```
 ipv6 access-list web-test
 permit tcp host 2001:660:3203:430::A16 any  eq www
 permit tcp host 2001:660:3203:430::A21 any  eq www
```

Après nous avons défini un "route-map" qui vérifie sur l'adresse IP et le port de l'hôte est sur cette liste d'accès et qui va acheminer le trafic web vers le proxy

```
 route-map web-test-ipv6 permit 10
 match ipv6 address web-test
 set ipv6 next-hop 2001:660:3203:422::A99
```

Une fois ces règles mises en place, nous l'avons appliqué sur l'interface VLAN 430. Malgré cette configuration nous n'avons pas vu de paquets filtrés par la liste d'accès. Nous n'avons pas réussi à trouver la raison de ce problème. Finalement nous avons renoncé de mettre en place le proxy et nous avons utilisé une solution alternative pour surveiller le trafic web.

4.2.7 IDS

N'arrivant pas à faire fonctionner de proxy transparent sur notre réseau, nous voulions tout de même des outils permettant de diagnostiquer et si besoin de surveiller le trafic sur notre réseau. Nous avons pour cela mis en place un IDS sur une machine physique qui reçoit une copie du trafic entrant sur le pare-feu. Nous utilisons le logiciel Bro IDS. Une fois le paquet Debian disponible sur le site officiel installé, il suffit d'ajouter la ligne suivante à **/etc/rc.local** pour lancer le programme au démarrage :

```
 /usr/local/bro/bin/broctl start
```

Les logs sont enregistrés dans **/usr/local/bro/logs**. Plutôt que de se concentrer sur des signatures comme snort, Bro IDS cherche les comportement anormaux d'un point de vue protocolaire. Pour l'instant, nous nous contentons des règles par défaut, compatibles IPv6.

4.2.8 Télévision

En IPv6, sur les switchs et routeurs cisco, l'igmp snooping est remplacé par le mld snooping[**?**]. Pour ne pas inonder le réseau de paquets liés à la télévision, il convient donc de configurer les routeurs et les switchs

Voici la configuration en IPv4 :

```
ip multicast-routing
ip igmp snooping last-member-query-interval 3000
ip igmp snooping vlan X mrouter learn cgmp
ip igmp snooping vlan X immediate-leave
ip igmp snooping vlan X last-member-query-interval 1000
ip igmp snooping vlan X immediate-leave
no ip igmp filter
```

et en IPv6 :

```
 ipv6 mld snooping vlan X immediate-leave
 ipv6 mld snooping
 ipv6 multicast-routing
 ipv6 multicast rpf use-bgp
```

Certains switchs ne supportent pas le mld snooping. Cependant l'ensemble des switchs déployés à MiNET le supporte.

Nous avons dû mettre à jour Mumudvb, la solution nous permettant à la fois de capturer et de diffuser nos flux vidéo, car l'IPv6 n'est supportée qu'à partir de la version 1.7. Le mode autoconfiguration fonctionne tout aussi bien en IPv6 qu'en IPv4.

Le mld snooping gère les abonnements aux groupes de multicast et a donc un impact direct sur les flux en multicast nécessaires pour l'IPv6 (RA, NA, etc.). Nous n'avons pas réussi concilier au niveau du mld snooping les flux de télévision et les autres flux, l'établissement du mld bloquant systématiquement la totalité des flux multicasts. La RFC 3590 est dédiée à ce problème, son étude permettrait peut-être de le résoudre. En effet, à la première connexion, les clients ont besoin d'informations sur le réseau obtenues par multicast (avant l'obtention d'une adresse ip). Cependant, à ce stade, les clients ne peuvent se joindre aux groupes et ne reçoivent donc pas les messages en multicast nécessaires à l'établissement de la connexion. Nous avons donc dû supprimer le mld snooping.

Nous avons finalement décidé de ne pas diffuser la télévision en IPv6 pour les raisons suivantes :
- Les clients ne gèrant pas tous l'IPv6, nous sommes obligés de conserver la diffusion en IPv4.
- La solution de mld snooping n'étant pas opérationnelle, nous ne pouvons nous permettre d'inonder les adhérents avec les flux de télévision.
- Ne pas doubler le trafic lié à la télévision a un impact important sur les débits du réseau.

4.2.9 NTP

Le démon ntpd supporte nativement l'ipv6, il suffit de configurer une interface en ipv6 et de relancer le démon pour que l'ipv6 soit pris en compte.

4.3 Migration des adhérents

4.3.1 Modification des outils existants

ADH5

ADH5 est notre logiciel de gestion des adhérents codé avec le framework Ruby on Rails. Jusqu'à présent un VLAN était une plage d'IPv4 /24. Nous avons amélioré cela en caractérisant un VLAN par un numéro, une plage d'adresses IPv4 avec son masque et une plage d'adresses IPv6 avec son masque.

Nous avons pu repérer que le parsing des adresses IPv6 est très délicat.

Aussi, nous avons dû réaliser de nombreux tests avant de déployer la nouvelle version en production car cette interface répond aux requêtes Radius et DHCP à la fois en IPv4 et en IPv6. Un bug aurait pu affecter la partie IPv4.

DHCP

Nous avons dû modifier le patch existant appliqué à isc-dhcp pour permettre de récupérer l'adresse IP en fonction de l'adresse MAC par une requête Web sur ADH5, avec une réponse en JSON. Les adresses sont du type :

```
http://192.168.102.135:8081/dhcp/options[_ipv6]?mac={{mac address}}
```

Bien que beaucoup de code soit différent pour l'IPv6 dans isc-dhcp, la modification est située dans une portion commune. Nous avons définit dans un nouveau fichier server/remotedb.c une fonction find_haddr_with_remotedb, pour l'utiliser dans server/mdb.c, plus précisément dans la fonction find_hosts_by_haddr.

Pour des raisons de simplification des serveurs, notre dhcp pour l'ensemble des vlans des adhérents n'a qu'une seule interface dans le vlan des serveurs de production. Nous avons mis en place un relai dhcp sur notre routeur dans l'interface de chaque vlan.

```
ipv6 dhcp relay destination 2001:660:3203:422::A10
```

Ainsi les requêtes dhcp de chaque vlan sont transmises au dhcp dans un autre vlan par le routeur qui retransmet la réponse.

4.3.2 Migrations des adhérents

Pour la migration des adhérents, nous avons d'abord rempli la base de données des adresses ipv6 puis nous avons simplement activé l'ipv6, le dhcp-relay et les RA sur le routeur. Pour la migration des adhérents, nous avons d'abord rempli la base de données des adresses IPv6 puis nous avons simplement activé l'IPv6, le dhcp-relay et les RA sur le routeur. Nous avons commencé par le bâtiment U4 (60-70 personnes), puis une semaine après, n'ayant pas rencontré de problème, nous avons fait la même chose pour le reste des adhérents (600 personnes).

5. Conclusion

Le déploiement de l'IPv6 sur notre réseau a été un gros projet de plus de 400 heures de travail. Nous pensons avoir choisi la stratégie de déploiement optimale pour nous. La configuration des services en IPv6 à été l'occasion de mieux comprendre le fonctionnement de services en IPv4 et de nettoyer et améliorer les configurations IPv4 au passage. Nous avons acquéri des connaissances et de la pratique sur l'IPv6. Côté adhérents nous n'avons pas de problèmes, les implémentations dans les principaux OS semblent matures. Nous pensons que le passage à l'IPv6 est l'occasion pour nous d'améliorer encore l'image de l'association, d'acquérir des connaissances et de la pratique qui seront très utiles dans les années à venir et de résoudre des problèmes touchant à la fois nos services (manque d'IP) et nos adhérents (tunnels IPv6-IPv4). En guise de perspective, il serait intéressant de récolter des statistiques sur l'utilisation de l'IPv6 sur notre réseau et ses performances réelles par rapport à l'IPv4.

A. /etc/init.d/firewallv6

```
#!/bin/bash
#
# Script de lancement du firewall IPv6
# Pour toutes infos, netantho@minet.net ou mael@minet.net
#
# F vrier-Mars 2013

IPT=ip6tables
TC=/sbin/tc
LOCK=/var/lock/firewallv6.lock
BLACKLIST=/etc/iptables/ipv6_blacklist
LO=lo
IP_ADMIN=[IP administration pare-feu]
IF_ADMIN=eth0
IF_DISI=eth3
IF_MINET=eth2
IF_IDS=eth1
ROUTER_MINET="2001:660:3203:401::1 2001:660:3203:402::1 2001:660:3203:403::1
    2001:660:3203:404::1 2001:660:3203:405::1 2001:660:3203:406::1
    2001:660:3203:40F::1 2001:660:3203:411::1 2001:660:3203:412::1
    2001:660:3203:413::1 2001:660:3203:414::1 2001:660:3203:415::1
    2001:660:3203:416::1 2001:660:3203:41F::1 2001:660:3203:430::1"
NETWORK_DISI="2001:660:3203::/48"
NETWORK_MINET="2001:660:3203:400::/57"
NETWORK_MINET_SERVICES="2001:660:3203:420::/64 2001:660:3203:421::/64
    2001:660:3203:422::/64 2001:660:3203:423::/64 2001:660:3203:424::/64
    2001:660:3203:425::/64"
NETWORK_MINET_ENDUSERS="2001:660:3203:401::/64 2001:660:3203:402::/64
    2001:660:3203:403::/64 2001:660:3203:404::/64 2001:660:3203:405::/64
    2001:660:3203:406::/64 2001:660:3203:40F::/64 2001:660:3203:411::/64
    2001:660:3203:412::/64 2001:660:3203:413::/64 2001:660:3203:414::/64
    2001:660:3203:415::/64 2001:660:3203:416::/64 2001:660:3203:41F::/64
    2001:660:3203:430::/64"
PROXY_MINET="2001:660:3203:422::a81"
TEE_GATEWAY="fc00:8:8::a2"

DNS_MINET="2001:660:3203:422::a55 2001:660:3203:422::a54"
DNS_INT="157.159.10.12 157.159.10.13"
CHAINS="icmp P2P banned p2p-set p2p-set-bt p2p-set-ed2k p2p-set-gnutella minet
    -in minet-out portp2p adh-out adh-in portscan blacklist DoS ingress
    outbound"

do_start() {
  echo "starting..."

    [ ! -f "$BLACKLIST" ] && { echo "$0: File $BLACKLIST not found."; exit 1;
       }

    # Create chains
    echo "Create $CHAINS chains"
    for chain in $CHAINS
    do
        $IPT -N $chain
    done

    # Whitelist for input and forward
    # Blacklist for output
```

```
    echo "Applying␣default␣policies"
    $IPT -P INPUT DROP
    $IPT -P FORWARD DROP
    $IPT -P OUTPUT ACCEPT

    ##############################
    # Firewall's own protection #
    ##############################

    echo "Applying␣firewall's␣own␣protection"

    $IPT -A INPUT -p all -m conntrack --ctstate ESTABLISHED,RELATED -m comment
        --comment "Accept␣connexions␣already␣established" -j ACCEPT

    # Unlock loopback
    $IPT -A INPUT -i $LO -m comment --comment "Unlock␣loopback" -j ACCEPT

    # We don't accept anything else since we connect
    # to the firewall to manage it in ipv4

    ##############################
    # Firewall for the network #
    ##############################

    echo "Applying␣network␣protection"

    $IPT -A FORWARD -p all -m conntrack --ctstate ESTABLISHED,RELATED -m
       comment --comment "Accept␣connexions␣already␣established" -j ACCEPT

    # ICMP #
    ########

    # RFC 4890: http://www.ietf.org/rfc/rfc4890.txt
    # TODO RFC 4890 compliant
    # http://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xml
    # 1: Desination unreachable RFC4443. Authorized everywhere
    # 2: Packet Too Big RFC4443. Authorized everywhere
    # 3: Time Exceeded RFC4443. Authorized everywhere
    # 4: Parameter Problem RFC4443. Authorized everywhere
    # 128, 129: Echo request, echo reply RFC4443. authorized
    # 130-132 RFC 2710, 143 RFC3810, 151-153 RFC4286: Multicast. Authorized on
        LAN
    # 133-136: RS,RA,NS,NA RFC4861. Authorized only on LAN
    # 138: Router Renumbering. Block
    # 139-140: ICMP Node Information RFC4620. Block
    # 141-142: Inverse Neighbor Discovery RFC3122. Authorized only on LAN
    # 144-147: Mobility RFC6275. Block
    # 148-149: Certification Path Solicitation and Advertisement RFC3971.
       Authorized only on LAN
    # 154-199: block
    # 202-254: block
    # Default: block
    $IPT -A FORWARD -p icmpv6 -j icmp

    $IPT -A icmp -p icmpv6 --icmpv6-type destination-unreachable -j ACCEPT
    $IPT -A icmp -p icmpv6 --icmpv6-type packet-too-big -j ACCEPT
    $IPT -A icmp -p icmpv6 --icmpv6-type time-exceeded -j ACCEPT
    $IPT -A icmp -p icmpv6 --icmpv6-type parameter-problem -j ACCEPT
    $IPT -A icmp -p icmpv6 --icmpv6-type echo-request -j ACCEPT
```

```
    $IPT -A icmp -p icmpv6 --icmpv6-type echo-reply -j ACCEPT
    $IPT -A icmp -s $NETWORK_MINET -d $NETWORK_MINET -p icmpv6 --icmpv6-type
       130 -m comment --comment "ICMP␣Multicast:␣ICMPv6MLQuery" -j ACCEPT
    $IPT -A icmp -s $NETWORK_MINET -d $NETWORK_MINET -p icmpv6 --icmpv6-type
       131 -m comment --comment "ICMP␣Multicast:␣ICMPv6MLReport" -j ACCEPT
    $IPT -A icmp -s $NETWORK_MINET -d $NETWORK_MINET -p icmpv6 --icmpv6-type
       132 -m comment --comment "ICMP␣Multicast:␣ICMPv6MLDone" -j ACCEPT
    $IPT -A icmp -s $NETWORK_MINET -d $NETWORK_MINET -p icmpv6 --icmpv6-type
       143 -m comment --comment "ICMP␣Multicast:␣RFC␣3810" -j ACCEPT
    $IPT -A icmp -s $NETWORK_MINET -d $NETWORK_MINET -p icmpv6 --icmpv6-type
       151 -m comment --comment "ICMP␣Multicast:␣ICMPv6MRD_Advertisement" -j
       ACCEPT
    $IPT -A icmp -s $NETWORK_MINET -d $NETWORK_MINET -p icmpv6 --icmpv6-type
       152 -m comment --comment "ICMP␣Multicast:␣ICMPv6MRD_Solicitation" -j
       ACCEPT
    $IPT -A icmp -s $NETWORK_MINET -d $NETWORK_MINET -p icmpv6 --icmpv6-type
       153 -m comment --comment "ICMP␣Multicast:␣ICMPv6MRD_Termination" -j
       ACCEPT
    # We drop external and rogue internal RAs
    for router in $ROUTER_MINET
    do
        $IPT -A icmp -s $router -d $NETWORK_MINET -p icmpv6 --icmpv6-type
           router-advertisement -m hl --hl-eq 255 -j ACCEPT
    done
    $IPT -A icmp -s $NETWORK_MINET -p icmpv6 --icmpv6-type router-solicitation
        -m hl --hl-eq 255 -j ACCEPT
    $IPT -A icmp -s fe80::/10 -d ff00::/8 -p icmpv6 --icmpv6-type neighbor-
       solicitation -j ACCEPT
    $IPT -A icmp -s $NETWORK_MINET -d ff00::/8 -p icmpv6 --icmpv6-type
       neighbor-solicitation -j ACCEPT
    $IPT -A icmp -s fe80::/8 -d $NETWORK_MINET -p icmpv6 --icmpv6-type
       neighbor-solicitation -j ACCEPT
    $IPT -A icmp -s fe80::/8 -d fe80::/8 -p icmpv6 --icmpv6-type neighbor-
       solicitation -j ACCEPT
    $IPT -A icmp -s fe80::/10 -d fe80::/10 -p icmpv6 --icmpv6-type neighbor-
       advertisement -j ACCEPT
    $IPT -A icmp -s fe80::/10 -d $NETWORK_MINET -p icmpv6 --icmpv6-type
       neighbor-advertisement -j ACCEPT
    $IPT -A icmp -s $NETWORK_MINET -d fe80::/10 -p icmpv6 --icmpv6-type
       neighbor-advertisement -j ACCEPT
    $IPT -A icmp -s $NETWORK_MINET -d $NETWORK_MINET -p icmpv6 --icmpv6-type
       141 -m comment --comment "Inverse␣Neighbor␣Discovery␣RFC3122" -j ACCEPT
    $IPT -A icmp -s $NETWORK_MINET -d $NETWORK_MINET -p icmpv6 --icmpv6-type
       142 -m comment --comment "Inverse␣Neighbor␣Discovery␣RFC3122" -j ACCEPT
    $IPT -A icmp -s $NETWORK_MINET -d $NETWORK_MINET -p icmpv6 --icmpv6-type
       148 -m comment --comment "Certification␣Path␣Solicitation␣and␣
       Advertisement␣RFC3971" -j ACCEPT
    $IPT -A icmp -s $NETWORK_MINET -d $NETWORK_MINET -p icmpv6 --icmpv6-type
       149 -m comment --comment "Certification␣Path␣Solicitation␣and␣
       Advertisement␣RFC3971" -j ACCEPT

    $IPT -A icmp -j DROP

    # Network openness #
    ####################

    # NB: Do NOT BLOCK ripng trafic (very difficult to troubleshoot)
    $IPT -A FORWARD -s fe80::/10 -d ff00::/8 -p udp --dport  521 --sport 521 -
       m comment --comment "ripng␣between␣MiNET␣and␣DISI" -j ACCEPT
```

```
$IPT -A FORWARD -s $NETWORK_MINET -d $NETWORK_MINET -m comment --comment "
    Open traffic inside MiNET's network" -j ACCEPT

$IPT -A FORWARD -s $NETWORK_DISI -d $NETWORK_DISI -m comment --comment "
    Open traffic inside DISI's network" -j ACCEPT

# Denial of Service #
#####################

$IPT -A FORWARD -j DoS

$IPT -A DoS -m conntrack --ctstate INVALID -m comment --comment "Drop
    invalid states" -j DROP
# SYN and RST flood attacks should be detected with an IDS for us
# and rules pushed here if needed

$IPT -A FORWARD -m recent --name scanBanned --rcheck --seconds 3600 -m
    comment --comment "Protect against port scanning" -j DROP

# Ingress filtering (from outside to inside) #
##############################################

$IPT -A FORWARD -j ingress

# TODO RPFILTER
# Need Linux Kernel >= 3.3
# http://www.shorewall.net/Anti-Spoofing.html
# https://dev.openwrt.org/ticket/11272

# Implemented in network equipments, but just for redundancy
# unspecified ::/128
# multicast ff00::/8
# link-local unicast fe80::/10
# site-local unicast fec0::/48
# unique local unicast fc00::/7
$IPT -A ingress -s ::/128 -m comment --comment "Ingress Filtering:
    unspecified ::/128" -j DROP
$IPT -A ingress -s ff00::/8 -m comment --comment "Ingress Filtering:
    multicast ff00/8" -j DROP
$IPT -A ingress -s fe80::/10 -m comment --comment "Ingress Filtering: link
    -local fe80::/10" -j DROP
$IPT -A ingress -s fec0::/48 -m comment --comment "Ingress Filtering: site
    -local fec0::/48" -j DROP
$IPT -A ingress -s fc00::/7 -m comment --comment "Ingress Filtering:
    unique local unicast fc00::/7" -j DROP

# Outbound filtering (from inside to outside) #
###############################################

$IPT -A FORWARD -j outbound

# Tunnels
#   2002::/16 6to4 Tunneling
#   RFC 1933/2893 configured and automatic tunnels: IPv4.Protocol == 41
#   RFC 2529 6over4 tunnel: IPv4
#   RFC 3056 6to4 tunnel IPv4.Protocol == 41 && IPv4.{src,dst} ==
    192.88.99.0/24
#   ISATAP tunnel
```

```
    #   MobileIPv6 RFC 2473
    #   Teredo tunnels IPv4.dst == known_teredo_servers && UDP.DstPort == 3544
    #   6in4 IPv4.Protocol == 41
    #   6rd
    #   TSP IPv4.dst == known_teredo_servers && UDP.DstPort == 3653
    # Multicast ff00::/8
    # link-local unicast fe80::/10
    # site-local unicast fec0::/48
    # unique local unicast fc00::/7
    $IPT -A outbound -d 2002::/16 -m comment --comment "Outbound Filtering: 6
       to4 Tunneling" -j DROP
    $IPT -A outbound -s $NETWORK_MINET -p tcp --dport 3544 -m comment --
       comment "Outbound Filtering: Teredo Tunneling" -j DROP
    $IPT -A outbound -s $NETWORK_MINET -p udp --dport 3544 -m comment --
       comment "Outbound Filtering: Teredo Tunneling" -j DROP
    $IPT -A outbound -s $NETWORK_MINET -p tcp --dport 3653 -m comment --
       comment "Outbound Filtering: TSP Tunneling" -j DROP
    $IPT -A outbound -s $NETWORK_MINET -p udp --dport 3653 -m comment --
       comment "Outbound Filtering: TSP Tunneling" -j DROP

    # Blacklist #
    #############

    $IPT -A FORWARD -j blacklist

    CURRENT_RULES=`iptables -nL`
    while read entries ;do
        # skip comment lines starting with ; or #
        case $entries in
            \#*|\;*)
            continue
            ;;
        esac

        if [[ $CURRENT_RULES =~ $entries ]]
        then
            printf "%-20s %20s\n" $entries 'already referenced in ip6tables -
               skipping'
        else
            printf "%-20s %20s %1s %1s\n" 'ADDING RULE:' 'ip6tables -A
               FIREWALL -s' $entries '-j DROP'
            $IPT -A blacklist -s $entries -j DROP
        fi
    done < $BLACKLIST

    $IPT -A blacklist -j RETURN

    # Services
    for network in $NETWORK_MINET_SERVICES
    do
        $IPT -A FORWARD -s $network -j minet-out
        $IPT -A FORWARD -d $network -j minet-in
    done

    # End Users #
    #############

    for network in $NETWORK_MINET_ENDUSERS
    do
```

```
        $IPT -A FORWARD -s $network -j adh-out
        $IPT -A FORWARD -d $network -j adh-in
    done

    $IPT -A adh-out -j banned
    $IPT -A adh-out -j P2P
    $IPT -A adh-out -j ACCEPT

    $IPT -A adh-in -j banned
    $IPT -A adh-in -j P2P
    $IPT -A adh-in -j ACCEPT

    # Block ports <= 2000
    # DISI's rule
    $IPT -A adh-in -p tcp --dport 1:2000 -j DROP
    $IPT -A adh-in -p udp --dport 1:2000 -j DROP

    # 135: RPC
    # 137: Netbios Name Service
    # 138: Netbios Datagram service
    # 139: Netbios Session Service
    # 445: microsoft-ds (also  used by conficker & Zotob)
    # 1025: RPC, nfs, trojans
    # 3450: CAStorProxy, likely to have trojans listening
    # 5000: upnp (and a some trojans)
    # 5554: Sasser worm
    # 9996: Sasser worm as well
    $IPT -A banned -m multiport -p tcp --ports 135,1025 -m comment --comment "
       SecAss: RPC" -j DROP
    $IPT -A banned -m multiport -p udp --ports 135,1025 -m comment --comment "
       SecAss: RPC" -j DROP
    $IPT -A banned -m multiport -p tcp --ports 137,138,139 -m comment --
       comment "SecAss: Netbios" -j DROP
    $IPT -A banned -m multiport -p udp --ports 137,138,139 -m comment --
       comment "SecAss: Netbios" -j DROP
    $IPT -A banned -m multiport -p tcp --ports 445 -m comment --comment "
       microsoft-ds" -j DROP
    $IPT -A banned -m multiport -p udp --ports 445 -m comment --comment "
       microsoft-ds" -j DROP
    $IPT -A banned -m multiport -p tcp --ports 3450 -m comment --comment "
       SecAss: CAStroProxy and trojans" -j DROP
    $IPT -A banned -m multiport -p udp --ports 3450 -m comment --comment "
       SecAss: CAStroProxy and trojans" -j DROP
    $IPT -A banned -m multiport -p tcp --ports 5000 -m comment --comment "
       SecAss: UPNP and trojans" -j DROP
    $IPT -A banned -m multiport -p udp --ports 5000 -m comment --comment "
       SecAss: UPNP and trojans" -j DROP
    $IPT -A banned -m multiport -p tcp --ports 5554,9996 -m comment --comment
       "SecAss: Sasser Worm" -j DROP
    $IPT -A banned -m multiport -p udp --ports 5554,9996 -m comment --comment
       "SecAss: UPNP and trojans" -j DROP

    $IPT -A banned -j RETURN

    # Servers #
    ############

    # Block ports < 2000
    # DISI's rule
```

```
$IPT -A minet-in -p tcp --dport 1:2000 -j DROP
$IPT -A minet-in -p udp --dport 1:2000 -j DROP

$IPT -A minet-in -p tcp --dport 22 -m comment --comment "Block␣SSH␣from␣
   outside" -j DROP

# $IPT -A minet-out -s 2001:660:3203:422::a104 -p udp --dport 1234 -m
   comment --comment "Block MiNET TV Phoenix" -j DROP
# $IPT -A minet-out -s 2001:660:3203:422::a108 -p udp --dport 1234 -m
   comment --comment "Block MiNET TV Hallali" -j DROP

$IPT -A minet-in -j portscan
$IPT -A minet-in -j ACCEPT
$IPT -A minet-out -j ACCEPT

# Port Scan #
#############

$IPT -A portscan -p tcp ! --syn -m conntrack --ctstate NEW -m comment --
   comment "Drop␣new␣TCP␣connexion␣without␣SYN" -j DROP
$IPT -A portscan -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG FIN,PSH
   ,URG -m limit --limit 30/min -j RETURN
$IPT -A portscan -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG FIN,PSH
   ,URG -j DROP
$IPT -A portscan -p tcp -m tcp --tcp-flags FIN,SYN FIN,SYN -m limit --
   limit 30/min -j RETURN
$IPT -A portscan -p tcp -m tcp --tcp-flags FIN,SYN FIN,SYN -j DROP
$IPT -A portscan -p tcp -m tcp --tcp-flags SYN,RST SYN,RST -m limit --
   limit 30/min -j RETURN
$IPT -A portscan -p tcp -m tcp --tcp-flags SYN,RST SYN,RST -j DROP
$IPT -A portscan -p tcp -m tcp --tcp-flags RST RST,ACK -m limit --limit 5/
   min -j RETURN
$IPT -A portscan -p tcp -m tcp --tcp-flags RST RST,ACK -j DROP

#    P2P   #
############

$IPT -A P2P -m recent --name p2pLimit --rcheck --seconds 300 -j MARK --set
   -mark 104
$IPT -A P2P -j portp2p

$IPT -A P2P -j RETURN

# Also block common P2P ports
$IPT -A portp2p -p udp -m multiport --ports 1214,3531,4662,6257,41170 -j
   p2p-set
$IPT -A portp2p -p tcp -m multiport --ports
   412,1214,2234,2240,2834,3531,4661:4665,4672 -j p2p-set
$IPT -A portp2p -j RETURN

# TEE to IDS
$IPT -A INPUT -i $IF_IDS -m comment --comment "Unlock␣IDS␣interface" -j
   ACCEPT
$IPT -A PREROUTING -t mangle -i br0 -j TEE --gateway $TEE_GATEWAY

#############
#    QOS    #
#############
```

```
    echo "QOS"

    # To the Internet

    # 4 classes
    # Class 1 is the highest priority and 4 is the lowest
    # Bandwith limitation for class 4
    $TC qdisc add dev $IF_DISI root handle 1: prio bands 4
    $TC qdisc add dev $IF_DISI parent 1:1 handle 10: pfifo
    $TC qdisc add dev $IF_DISI parent 1:2 handle 20: pfifo
    $TC qdisc add dev $IF_DISI parent 1:3 handle 30: pfifo
    $TC qdisc add dev $IF_DISI parent 1:4 handle 40: tbf rate 100kbit buffer
       10000 latency 1s

    # packets with 104 mark is in class 4 (P2P)
    $TC filter add dev $IF_DISI parent 1:0 protocol ip handle 104 fw flowid
       1:4

    # To MiNET
    $TC qdisc add dev $IF_MINET root handle 1: prio bands 4
    $TC qdisc add dev $IF_MINET parent 1:1 handle 10: pfifo
    $TC qdisc add dev $IF_MINET parent 1:2 handle 20: pfifo
    $TC qdisc add dev $IF_MINET parent 1:3 handle 30: pfifo
    $TC qdisc add dev $IF_MINET parent 1:4 handle 40: tbf rate 100kbit buffer
       10000 latency 1s
    $TC filter add dev $IF_MINET parent 1:0 protocol ip handle 104 fw flowid
       1:4

    echo "OK"

    touch $LOCK
}

do_stop() {
  echo "stopping..."

    # On flush les cha nes par d faut
    $IPT -F INPUT
    $IPT -F OUTPUT
    $IPT -F FORWARD

    # On remet les politiques pour accepter tout
    $IPT -P INPUT ACCEPT
    $IPT -P OUTPUT ACCEPT
    $IPT -P FORWARD ACCEPT

    # On flush et supprime les cha nes mangle et filter
    $IPT -t mangle -F
    $IPT -t mangle -X
    $IPT -t filter -F
    $IPT -t filter -X

    # On supprime la QoS
    $TC qdisc del dev $IF_MINET root
    $TC qdisc del dev $IF_DISI root

    rm $LOCK
}
```

```
case "$1" in
  start)
  if [ -f $LOCK ]
  then
  echo "firewallv6 already launched" >&2
  else
  do_start
  fi
  ;;
  stop)
  if [ ! -f $LOCK ]
        then
        echo "firewallv6 not running" >&2
        else
        do_stop
        fi
  ;;
  status)
  if [ -f $LOCK ]
        then
        echo "firewallv6 running" >&2
        else
        echo "firewallv6 not running" >&2
        fi
  ;;
  restart|force-reload)
  if [ -f $LOCK ]
        then
        do_stop
        fi
  do_start
  ;;
  *)
  echo "Usage: firewallv6 {start|stop|status|restart|force-reload}" >&2
        exit 3
  ;;
esac
exit 0
```

Bibliographie

[1] J. Rosenwald H. Lord, M. O'Reirdan. Recommendations for the transition of email services from ipv4 to ipv6 for internet service providers. http://tools.ietf.org/html/draft-oreirdan-rosenwald-ipv6mail-transition-01, June 2013.

[2] Olle E. Johansson. Ipv6 and e-mail spam – we need to find new ways to block the bad guys ! http://ipv6friday.org/blog/2012/10/ipv6-spam/, October 2012.

[3] Wietse Venema. Postfix ipv6 support. http://www.postfix.org/IPV6_README.html, June 2013.