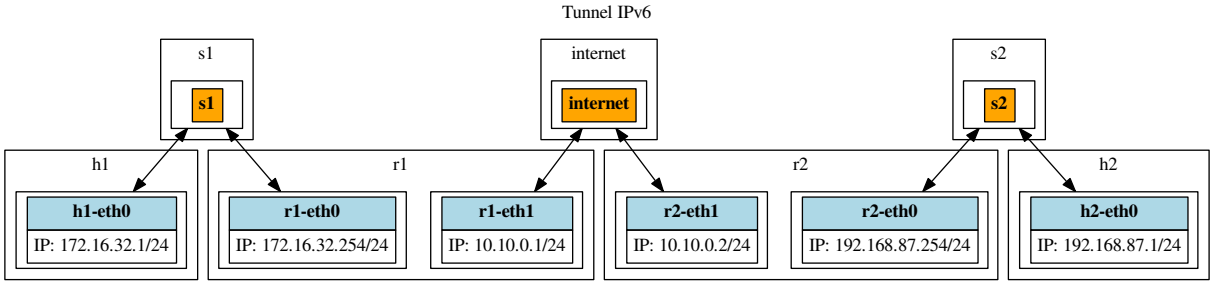
IPv6 Tunnel

Utilisation et mise en place sécurisé de Tunnel dynamique IPv4overIPv6

Le but est d'utiliser IPv6 pour créer un lien de communication dynamique entre deux réseaux privés.

Vous récupérez la configuration du réseau :

```
xterm
$ git clone https://git.p-fb.net/pef/ipv6_lab.git
```



- ▷ *un utilisateur dispose de l'hôte 1, « `h1` », et l'autre utilisateur dispose de l'hôte 2, « `h2` » ;*
- ▷ *chaque hôte est connecté dans un réseau privé dans les switches « `s1` » et « `s2` » respectivement ;*
- ▷ *dans chaque réseau privé, un routeur est connecté. Les deux routeurs r1 et r2 sont connectés entre eux dans un troisième réseau dans le switch « `internet` ».*

Les hôtes et les routeurs seront simulés à l'aide des « netns *».*

Présentation de la procédure de mise en place du tunnel :

Les deux utilisateurs doivent tout à bord se découvrir de manière sécurisée dans le résau local : on va créer un identifiant partagée sécurisé pour définir sa propre adresse IPv6, suivant la procédure suivante :

- ⋆ lorsque deux utilisateurs veulent se « trouver » dans le réseau local :
  - ◇ ils conviennent d'un secret partagé S ;
  - ◇ ils utilisent la fonction de hashage MD5 pour obtenir un haché sous forme hexadécimal de ce secret partagé :

```
xterm
$ echo  'mon_secret' | md5sum
103053296d2c07679cf247ae82ee8d78  -
```

Attention : le echo envoie un « `\n` » à la fin du texte. Il faut utiliser « `echo -n 'toto'` » sinon.

  - ◇ ils récupèrent les 6 premières valeurs hexadécimale : *Ici,* `103053296d2c`
- ⋆ ils construisent une adresse IPv6 de lien local, `FE80::/64` avec ce préfixe et leur EUI :
  - ◇ obtention de l'@MAC :

```
xterm
$ sudo ip netns exec r1 bash
# ip link show
2: r1-eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
UP mode DEFAULT qlen 1000
    link/ether 00:11:de:ad:be:ef brd ff:ff:ff:ff:ff:ff
```

  - ◇ construction d'une adresse de lien : `fe80:1030:5329:6d2c:211:deff:fead:beef/64`
  - ◇ configuration de l'interface :

```
xterm
# sudo ip addr add fe80:1030:5329:6d2c:211:deff:fead:beef/64 dev r1-eth1
# ip addr show dev r1-eth1
  ...
  inet6 fe80:1030:5329:6d2c:211:deff:fead:beef/64 scope link
```

- on vérifie la configuration automatique de la table de routage pour IPv6 :

```
xterm
# ip -6 route
fe80::/64 dev r1-eth0 proto kernel metric 256  pref medium
fe80::/64 dev r1-eth1 proto kernel metric 256  pref medium
fe80:1030:5329:6d2c::/64 dev r1-eth1 proto kernel metric 256  pref medium
```

⋆ ils se font connaître sur le réseau local de manière « discrète » :
- en utilisant la possibilité offerte à un routeur de faire de « l'advertisement » de préfixe réseau pour la construction d'adresse IPv6 globale, à l'aide du protocole IPv6 « *Neighbor Discovery* » et de l'option « *Router Advertisement* » ;
  Le routeur diffuse un préfixe global et chaque poste qui reçoit ce message peut construire sa propre adresse globale en même temps qu'il apprend l'adresse d'un roueur pour sortir de son réseau local.
- en diffusant un préfixe où ils vont mettre le haché du secret et leur @MAC pour prévenir de leur présence leur complice :
  `3001:1030:5329:6d2c::/64` dans le message « *Router Advertisement* »
  L'utilisateur pourra extraire l'adresse de son complice dans l'@IP d'origine du paquet.

⋆ une fois découvert cette diffusion dont il est le destinataire après comparaison avec le haché du secret qu'il possède lui aussi, l'autre utilisateur récupère l'@IPv6 pour joindre son complice ;

⋆ chaque utilisateur crée une adresse IPv6 globale avec un préfixe contenant également son haché et distinct de celle utilisée pour le « router advertisement » :
`3001:1030:5329:6d2c:211:deff:fead:beef/64` par exemple.

⋆ l'utilisateur mets alors en place un tunnel IPv4 over IPv6, c-à-d un tunnel où les paquets IPv4 circulent à l'intérieur d'IPv6 entre son réseau privé interne et le réseau privé de son complice, en utilisant les adresses globales en préfixe `3001:1030:5329:6d2c::/64` commes extrémités du tunnel.

⋆ les deux utilisateurs peuvent maintenant communiquer entre eux à partir de leur hôte appartenant à leur réseau rexpectif.

Tester la connectivité en IPv6 entre les routeurs

Avant de commencer la mise en place automatique de tunnel, vous pouvez vérifier que la connectivité en IPv6 fonctionne bien entre vos deux routeurs.
Si vous avez ajouté votre adresse en `fe80:1030:5329:6d2c::/64` sur r1 et que vous connaissez l'adresse IPv6 de r2 où vous aurez également ajouté l'adresse avec ce même préfixe :

```
xterm
$ ip netns exec r2 bash
# ip address
32: r2-eth1@if31: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default qlen 1000
    link/ether 2a:f1:f9:bb:98:9a brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.10.0.2/24 scope global r2-eth1
       valid_lft forever preferred_lft forever
    inet6 fe80:1030:5329:6d2c:28f1:f9ff:febb:989a/64 scope link
```

Vous pouvez utiliser un « ping » :

```
xterm
$ sudo ip netns exec r1 bash
# ping6 fe80:1030:5329:6d2c:28f1:f9ff:febb:989a%r1-eth1
```

On utilise l'adresse de r2

La notation de l'adresse avec « `%r1-eth1` » permet à la commande `ping6` de connaître l'interface à utiliser. Vous pourriez également utiliser à la place l'option « `-I r1-eth1` ».

Vous pouvez tester une connexion TCP à l'aide de la commande « `socat` » entre les deux netns, r1 et r2 :

```
xterm
$ sudo ip netns exec r1 bash
# socat tcp6-listen:8090 -
```

Sur le second routeur, r2 :

```
xterm
$ sudo ip netns exec r2 bash
# socat tcp6:[fe80:1030:5329:6d2c:211:deff:fead:beef%r2-eth1]:8090 -
```

On utilise l'adresse de r1

Pour isoler l'adresse IPv6 dans la commande « `socat` », il faut la noter entre « crochets ».

Diffusion du « *Router advertisement* » avec Scapy

Il est possible d'envoyer des paquets « d'advertisement » de préfixe global à l'aide de Scapy :

```
p=Ether()/IPv6(src="fe80:1030:5329:6d2c:28f1:f9ff:febb:989a")/
  ICMPv6ND_RA(routerlifetime=10)/
  ICMPv6NDOptPrefixInfo(prefix="3001:1030:5329:6d2c::", prefixlen=64,
  validlifetime=10,preferredlifetime=10)
> sendp(p)
```

- ▷ *on indiquera son @IPv6 avec le préfixe construit à partir du secret comme source du paquet IPv6 ;*
- ▷ *on diffusera un préfixe en « /64 » en accord avec la procédure ;*
- ▷ *on choisira une durée de vie courte pour ces informations.*
- ▷ *dans la construction du paquet, il est possible d'ajouter une information indiquant l'@MAC du routeur (util pour faire une attaque MITM en IPv6) :*

```
...,validlifetime=10,preferredlifetime=10)
  /ICMPv6NDOptSrcLLAddr(lladdr="00:b0:b0:67:89:AB")
```

Une fois le paquet envoyé, il est possible de voir sur l'autre machine l'effet produit :

xterm

```
# ip address show dev r1-eth1
28: r1-eth1@if27: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default qlen 1000
    link/ether a2:0b:01:19:ca:9d brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.10.0.1/24 scope global r1-eth1
       valid_lft forever preferred_lft forever
    inet6 fe80:1030:5329:6d2c:211:deff:fead:beef/64 scope link
       valid_lft forever preferred_lft forever
     ...
# ip -6 route
3001:1030:5329:6d2c::/64 dev r1-eth1  proto kernel  metric 256  expires 4sec
...
default via fe80:1030:5329:6d2c:28f1:f9ff:febb:989a dev r1-eth1  proto kernel
metric 1024  expires 4sec
```

Le poste se crée une @IPv6 avec le préfixe reçu, et la durée de vie de cette adresse, ainsi que de la route associée avec sa passerelle est $< 10s$.

Pour la réception de ce « *routeur advertisement* », vous aurez le choix entre surveiller la configuration de votre interface réseau ou bien de « sniffer » avec Scapy.

Configuration d'un tunnel « *IPv4 over IPv6* »

Pour mettre en place le tunnel « ip4ip6 » :

xterm

```
# ip -6 tunnel add <interface_virtuelle> mode ipip6 remote
<adresse_globale_ipv6_tunnel_distant> local <adresse_globale_ipv6_locale>
```

Exemple :

votre adresse en 3001::

xterm

```
# ip -6 tunnel add tnlipv6 mode ipip6 remote <adresse_globale_ipv6_complice> local
<adresse_globale_ipv6_locale>
# ip link set dev tnlipv6 up
# ip route add <reseau destination> dev tnlipv6
```

L'ajout de la route permet, aux paquets à destination du réseau privé de votre interlocuteur, d'emprunter le tunnel.

Pour afficher le tunnel :

```
$ ip -6 tunnel show mode any
```

Pour l'enlever :

```
# ip tunnel del tnlipv6
```

Sur r1 :

xterm

```
$ sudo ip netns exec r1
# ip address
28: r1-eth1@if27: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group
default qlen 1000
    link/ether a2:0b:01:19:ca:9d brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.10.0.1/24 scope global r1-eth1
       valid_lft forever preferred_lft forever
    inet6 3001:1030:5329:6d2c:211:deff:fead:beef/64 scope global
       valid_lft forever preferred_lft forever
# ip -6 tunnel add tnlipv6 mode ipip6 remote 3001:1030:5329:6d2c:28f1:f9ff:febb:989a local
3001:1030:5329:6d2c:211:deff:fead:beef
# ip l set dev tnlipv6 up
# ip route
10.10.0.0/24 dev r1-eth1 proto kernel scope link src 10.10.0.1
172.16.32.0/24 dev r1-eth0 proto kernel scope link src 172.16.32.254
192.168.87.0/24 via 10.10.0.2 dev r1-eth1
# ip route del 192.168.87.0/24 via 10.10.0.2 dev r1-eth1
# ip route add 192.168.87.0/24 dev tnlipv6
```

Sur r2 :

xterm

```
$ ip netns exec r2 bash
# ip address
32: r2-eth1@if31: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group
default qlen 1000
    link/ether 2a:f1:f9:bb:98:9a brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.10.0.2/24 scope global r2-eth1
       valid_lft forever preferred_lft forever
    inet6 3001:1030:5329:6d2c:28f1:f9ff:febb:989a/64 scope global
       valid_lft forever preferred_lft forever
# ip -6 tunnel add tnlipv6 mode ipip6 remote 3001:1030:5329:6d2c:211:deff:fead:beef local
3001:1030:5329:6d2c:28f1:f9ff:febb:989a
# ip l set tnlipv6 up
# ip route
10.10.0.0/24 dev r2-eth1 proto kernel scope link src 10.10.0.2
172.16.32.0/24 via 10.10.0.1 dev r2-eth1
192.168.87.0/24 dev r2-eth0 proto kernel scope link src 192.168.87.254
# ip route del 172.16.32.0/24 via 10.10.0.1 dev r2-eth1
# ip route add 172.16.32.0/24 dev tnlipv6
```

Sur h1 :

xterm

```
$ ip netns exec h1 bash
# ping -c 1 192.168.87.1
PING 192.168.87.1 (192.168.87.1) 56(84) bytes of data.
64 bytes from 192.168.87.1: icmp_seq=1 ttl=62 time=2.10 ms

--- 192.168.87.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 2.105/2.105/2.105/0.000 ms
```

Pour écouter sur r1 :

xterm

```
# tcpdump -lnvv -i r1-eth1 ip6
tcpdump: listening on r1-eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
02:00:07.117992 IP6 (flowlabel 0xd4f46, hlim 64, next-header unknown (60) payload length: 92)
3001:1030:5329:6d2c:211:deff:fead:beef > 3001:1030:5329:6d2c:28f1:f9ff:febb:989a: DSTOPT
(opt_type 0x04: len=1)(padn) IP (tos 0x0, ttl 63, id 31493, offset 0, flags [DF], proto ICMP
(1), length 84)
    172.16.32.1 > 192.168.87.1: ICMP echo request, id 11527, seq 1, length 64
02:00:07.119293 IP6 (flowlabel 0xd4f46, hlim 64, next-header unknown (60) payload length: 92)
3001:1030:5329:6d2c:28f1:f9ff:febb:989a > 3001:1030:5329:6d2c:211:deff:fead:beef: DSTOPT
(opt_type 0x04: len=1)(padn) IP (tos 0x0, ttl 63, id 4704, offset 0, flags [none], proto ICMP
(1), length 84)
    192.168.87.1 > 172.16.32.1: ICMP echo reply, id 11527, seq 1, length 64
```

1 – Analyse du tunnel :

a. Quelle est la MTU de l'interface correspondant au tunnel ?

b. Quelle est la valeur retournée sous Scapy par la commande :

```
len(str(IPv6ExtHdrFragment()))
```

Expliquez comment est déterminée cette MTU.

c. Vous sniffrez un paquet échangé dans le tunnel, en faisant par exemple un « ping » vers l'hôte A du réseau privé de votre interlocuteur.
Comment est « composé » votre tunnel ?

2 – Vous testerez :

a. l'envoi de paquet « Router Advertisement » construit suivant la procédure décrite (haché du secret, dérivation des adresses IPv6, *etc.*

b. la configuration de l'interface et de la table de routage sur la machine de votre interlocuteur.

Si le temps de vérification excède 10s la configuration de l'interface risque de disparaître.

3 – Écrivez deux outils permettant la mise en place automatique de ce tunnel suivant la procédure de découverte automatique :

a. en mode attente : il attend que votre interlocuteur envoie le message de « Router advertisement » et met en place le tunnel ;

b. en mode connexion : il envoie le « Router Advertisement », puis déclenche l'outil en mode attente.

Vous utiliserez Scapy pour l'injection des paquets « Router advertisement ».

Pour la récupération des paquets « Router Advertisement », vous pouvez utiliser Scapy ou bien surveiller la configuration de vos interfaces…

Configuration du firewall sur r1 & r2

4 – Vous commencerez par bloquer tout le trafic en entrée de l'hôte pour la pile tcp/ip en version 6 :

Sur r1 :

xterm

```
# ip6tables -t filter -P INPUT DROP
# ip6tables -t filter -nvL
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Vous ajouterez la règle permettant d'autoriser les paquets à destination de toutes les machines du réseau local, sur r1 :

xterm

```
# ip6tables -t filter -A INPUT -d ff02::1 -j ACCEPT
# ip6tables -t filter -nvL
Chain INPUT (policy DROP 14 packets, 3152 bytes)
 pkts bytes target     prot opt in     out     source               destination
    3   232 ACCEPT     all      *      *       ::/0                 ff02::1/128

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 32 packets, 4464 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

a. Vous vérifierez que l'injection du paquet « *router advertisement* » fonctionne toujours et ajoute toujours une adresse avec le préfixe indiqué.

b. Vous ajouterez des règles qui n'autoriserons que le transfert des paquets IPv6 à destination ou en provenance de votre tunnel.

Attention

Sur les dernières versions d'Ubuntu, il est nécessaire d'autoriser l'acceptation des « router advertisement » :

pef@solaris

```
sudo sysctl -w net.ipv6.conf.eth0.accept_ra=2
```