Les tunnels

Grâce à un tunnel, il est possible de passer directement d'un point à un autre, sans devoir subir les affres de la circulation à la surface. Les tunnels informatiques s'en rapprochent fortement, en proposant un moyen de relier "directement" deux réseaux privés distants, à travers un inter-réseau aussi complexe que l'internet.

Il existe une grande quantité de moyens pour réaliser des tunnels informatiques. PPP peut être considéré comme un tunnel dans des configurations comme PPPoE ou PPPoA. Ce sont des tunnels sur la couche 2 du modèle OSI, au même titre que L2TP (Layer 2 Tunneling Protocol), utilisé sur les réseaux des opérateurs, mis en oeuvre[1] dans les connexions ADSL non dégroupées.

Au niveau 3, il existe également plusieurs solutions, comme PPTP (Point to Point Tunneling Protocol), utilisé par Microsoft, ou encore les tunnels sur IPSec. L'objectif de ce chapitre est de monter le fonctionnement d'un tunnel sur IP à travers une implémentation standardisée : le tunnel GRE.

Merci à _SebF, créateur du site frameip.com[2], pour son aimable collaboration.

1 L'ADSL en pratiqaue : http://christian.caleca.free.fr/adsl/dans_la_pratique.htm
2 frameip.com : http://www.frameip.com/accueil/

Plan du chapitre

Le principe....3
Présentation générale....3
Comment ça marche....4
Description des réseaux....4
Réseau A....4
Réseau B....5
Démonstration....5
Tunnel GRE....6
Avertissements....6
Construction....6
Les deux réseaux....6
On creuse....6
Réseau A....7
Réseau B....7
Vérifications....8
Les interfaces....8
Snif d'un ping....9
Conclusions....10
Le protocole : que disent les RFCs ?....12
Utilisation....15
Premier exemple....15
Deuxième exemple....16
Troisième exemple....17
Limites....18
Conclusion....20

Le principe

Présentation générale

Imaginons que nous ayons à intervenir sur deux réseaux privés différents, géographiquement éloignés, les réseaux A et B. Si nous voulons interconnecter ces deux réseaux, nous avons à priori deux possibilités :

- l'une chère, qui consiste à utiliser une liaison spécialisée, proposée par tout bon opérateur de télécoms. Les technologies utilisées par ces opérateurs afin de créer notre réseau privé sont principalement du type ATM[3], MPLS[4] et, plus anciennement, Frame Relay.
  Les avantages apportés sont la garantie d'un SLA[5] et d'une étanchéité renforcée,
- l'autre, moins chère, qui consiste à interconnecter ces deux réseaux via de l'internet public.

Oui, mais la seconde solution, à priori moins chère, sera plus limitative.

- Soit, comme c'est le plus souvent le cas, nous ne disposerons que d'une seule IP publique pour accéder à chaque réseau et dans ce cas, nous ne pourrons pas faire facilement communiquer n'importe quelle machine du réseau A avec n'importe quelle machine du réseau B, puisque ces LANs seront montés avec des adresses IP privées. (Voyez le masquage d'adresses[6], mis en oeuvre dans de telles configurations),
- Soit nous disposons de suffisamment d'adresses IP publiques pour monter nos réseaux avec ces adresses, mais alors, toutes nos machines seront directement exposées sur le Net. Cher et difficile (il n'est pas simple, et encore moins gratuit d'obtenir des plages, même partielles, d'adresses IP publiques) et pour le moins dangereux.

Comment faire alors ?

Créer une ligne spécialisée virtuelle, qui passera par l'internet, mais qui fonctionnera presque comme une liaison spécialisée. Bien sûr, pour ce faire, un tunnel est nécessaire afin de créer l'interconnexion, de garantir l'étanchéité. L'avantage est de ne pas être dépendant d'un opérateur et ainsi, de pouvoir choisir la sortie Internet de chaque site indépendamment les unes des autres. Rien en effet n'interdit de construire plusieurs tunnels, éventuellement sur des connexions internet différentes. Nous disposons de plusieurs technologies telles que PPtP, IPSec et celle qui nous

3 Asynchronous Transfer Mode.
TTA en français (technologie temporelle asynchrone). Technologie de réseau à haut débit (centaines de Mbit/s, jusqu'à 2.5 Gbit/s) très sérieusement normalisée et présentée comme une solution d'avenir. Les données sont transmises sous forme de paquets de taille fixe (53 octets de long), appelés cellule. Originellement conçu pour la voix et les WANs, l'ATM est en train de passer aux LANs. Cette technologie semble prendre l'ascendant en ce moment sur ses concurrentes. Le principal problème étant la rapidité de commutation, qui doit être élevée vue la taille des cellules. Voir aussi "frame relay". En plus, c'est Made in France, mais personne ne le sait.

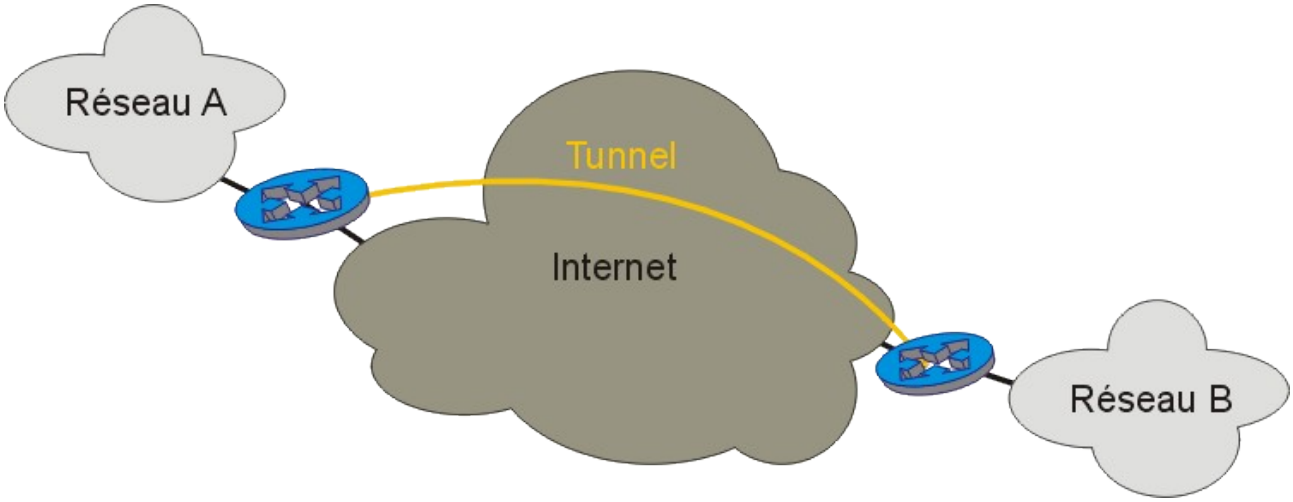
4 MultiProtocol Label Switching.
Technique conçue par l'IETF(http://www.ietf.org/) pour faire remonter des informations de niveau 2 OSI, comme la bande passante ou la latence d'un lien, dans la couche 3 (e.g. IP). Cela permet de gérer un peu mieux les ressources disponibles au sein d'un réseau.

5 Service Level Agreement.
Accord entre un client et un fournisseur sur le niveau de qualité de service offert par ce dernier. Le SLA est souvent suivi d'un SLM : Service Level Management. Gestion de la qualité de service, lors de laquelle on s'assure que le fournisseur tient bien ses promesses d'un point de vue qualitatif.

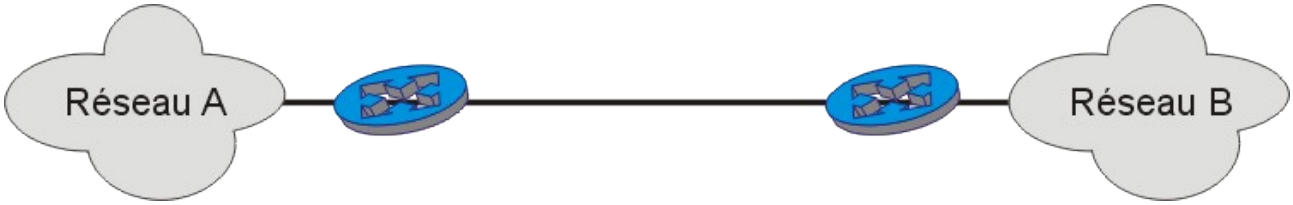
6 Masquerade : http://christian.caleca.free.fr/masquerade/

intéresse dans cette documentation : le tunnel GRE.

Au niveau IP, un tunnel se présente comme ceci :



Et nous aurons l'impression d'avoir à peu près cela :



Grâce à ce tunnel, tout noeud du réseau A pourra communiquer avec tout noeud du réseau B, les deux réseaux étant construits avec des IP **privées**, le tout passant dans le tunnel creusé dans l'internet.

Super non ?

Oui, mais souvenez-vous que IPv4 est un protocole qui n'est pas sécurisé, que nous allons l'utiliser et qui plus est, sur un réseau plutôt mal famé. L'opération n'est donc pas sans risques.

Comment ça marche

Toujours le même principe, l'encapsulation d'un protocole dans un autre protocole de même niveau. Le plus souvent, nous encapsulerons de l'IP dans de l'IP. Mais pour mieux comprendre, il nous faut poser le problème de façon plus précise.

Description des réseaux

Réseau A

Adresses :	172.16.0.0
Masque :	255.255.0.0
Routeur côté LAN :	172.16.254.1
Routeur côté Internet :	81.248.152.18

Réseau B

Adresses :	192.168.0.0
Masque :	255.255.255.0
Routeur côté LAN :	192.168.0.252
Routeur côté Internet :	80.8.147.232

Les routeurs A et B peuvent discuter entre eux à travers l'internet, puisqu'ils disposent tous deux d'une adresse IP publique. Au niveau IP, le transfert de données est donc réalisable.

Sur cette couche IP, nous encapsulons une seconde couche IP, qui va faire de telle sorte que l'ensemble des routeurs A et B avec le tunnel entre les deux, apparaisse comme un unique routeur, directement connecté aux réseau A et B, et qui aura :

- 172.16.254.1 dans le réseau A
- 192.168.0.252 dans le réseau B

Nous sommes entre deux réseaux privés, interconnectés par un routeur, rien que de bien classique.

Démonstration

Peu importe pour l'instant, la façon dont le tunnel est créé. Depuis un hôte de réseau B d'IP 192.168.0.10 , nous faisons un traceroute vers l'hôte du réseau A d'IP 172.16.252.2 :

```
C:\>tracert -d 172.16.252.2

Détermination de l'itinéraire vers 172.16.254.2 avec un maximum de 30 sauts.

  1    <1 ms    <1 ms    <1 ms  192.168.0.252
  2    60 ms    63 ms    59 ms  172.16.254.1
  3    75 ms    63 ms    61 ms  172.16.252.2

Itinéraire déterminé.
```

Et pourtant, si l'on cherche la "vraie route", celle qui est réellement empruntée par le tunnel, nous aurons quelque chose de cette forme entre les deux routeurs :

```
gw2:~# traceroute -n -I 81.248.152.18
traceroute to 81.248.152.18, 30 hops max, 38 byte packets
1 80.8.160.1 49.763 ms 32.095 ms 8.960 ms
2 172.19.46.65 10.883 ms 8.700 ms 11.688 ms
3 193.252.227.82 17.019 ms 23.244 ms 22.403 ms
4 80.10.209.233 38.494 ms 12.467 ms 12.732 ms
5 81.248.152.18 58.968 ms 60.676 ms 60.891 ms
```

Bien entendu, le "vrai" chemin peut être beaucoup plus long, si les deux réseaux A et B sont plus éloignés, mais le chemin par le tunnel gardera sa simplicité dans tous les cas.

Tunnel GRE

Avertissements

Il existe avec Linux un type de tunnel qui encapsule de l'IP sur IP. Cette solution n'existe, semble-t-il, que sous Linux et reste assez limitative. Nous allons plutôt utiliser une méthode normalisée, à peine plus complexe : le tunnel GRE. Ne confondez donc pas ces deux possibilités.

Le tunnel GRE (Generic Routine Encapsulation) fonctionne parfaitement. C'est un protocole ouvert, initialement développé par CISCO, et qui peut donc se mettre en place sur des plate-formes différentes. Il est défini par la RFC 2784[7] :

- il est possible d'ouvrir plusieurs tunnels depuis un hôte donné.
- il est conçu pour pouvoir encapsuler n'importe quel protocole de niveau 3 dans IP. Pratiquement, le plus souvent, de l'IP dans de l'IP.

Malheureusement, ce n'est pas un protocole sécurisé. Si vous l'utilisez sur l'internet, mesurez les risques que vous prenez !

Faites une recherche sur les façons de prendre possession d'un réseau utilisant un tel tunnel et vous serez fixé. Ce n'est pas facile à faire, mais c'est tout à fait réalisable. Vous êtes prévenu.

- GRE ne prévoit pas de chiffrment des données qui passent dans le tunnel, il n'est pas étanche,
- GRE ne prévoit pas l'authentification des extrémités du tunnel, vous n'êtes sûr que de l'authenticité de votre bout de tunnel.

(Je l'ai peut-être déjà dit...)

Construction

Juste pour voir comme c'est simple à monter, et comme un tunnel peut rendre des services, nous allons tout de même en réaliser un, le temps de faire la manip.

Reprenons la topologie utilisée :

Les deux réseaux

Les deux réseaux A et B sont les mêmes que définis sur la page précédente.

Les deux routeurs sont des machines sous Linux, avec un noyau 2.4.x. Il nous faut disposer d'iproute2. Toutes les distributions le proposent, mais ne l'installent pas forcément par défaut.

On creuse

Un tunnel, ça se creuse des deux côtés à la fois. Il faut donc intervenir sur les deux routeurs.

7 RFC 2784 : http://www.faqs.org/rfcs/rfc2784.html

Réseau A

- Amener les outils. Il faut charger le module nécessaire à la construction du tunnel :

```
modprobe ip_gre
```

- Construire le tunnel :

```
ip tunnel add netb mode gre remote 80.8.147.232 local 81.248.152.18 ttl 255
```

    - netb est le nom de la nouvelle interface réseau qui va conduire au réseau B,
    - l'adresse IP de l'autre bout du tunnel (remote) est 80.8.147.232,
    - l'adresse IP de ce bout-ci du tunnel (local) est 81.248.152.18,
    - on indique un ttl (time to live) maximum (255).

- Monter l'interface réseau :

```
ip link set netb up
```

- Lui donner une IP qui sera la même que celle de l'interface qui supporte le tunnel dans le réseau local :

```
ip addr add 172.16.254.1 dev netb
```

    - Ici il s'agit de 172.16.254.1

- Baliser la route vers le réseau B :

```
ip route add 192.168.0.0/24 dev netb
```

Et voilà. Le tunnel est creusé du côté A. Reste à refaire la même chose du côté B (aux adresses IP près).

Réseau B

- Amener les outils. Il faut charger le module nécessaire à la construction du tunnel :

```
modprobe ip_gre
```

- Construire le tunnel :

```
ip tunnel add neta mode gre remote 81.248.152.18 local 80.8.147.232 ttl 255
```

    - neta est le nom de la nouvelle interface réseau qui va conduire au réseau A,
    - l'adresse IP de l'autre bout du tunnel (remote) est 81.248.152.18,
    - l'adresse IP de ce bout-ci du tunnel (local) est 80.8.147.232,
    - on indique un ttl (time to live) maximum (255)

- Monter l'interface réseau :

```
ip link set neta up
```

- Lui donner une IP qui sera la même que celle de l'interface qui supporte le tunnel dans le réseau local :

```
ip addr add 192.168.0.252 dev neta
```

    - Ici il s'agit de 192.168.0.252

- Baliser la route vers le réseau A :

```
ip route add 172.16.0.0/16 dev neta
```

Et le tunnel est entièrement creusé et opérationnel. Bien entendu, il vous faudra ajouter dans les règles IPtables, ce qui est nécessaire pour que ça fonctionne. C'est à vous de voir en fonction de vos règles en place. On peut imaginer que ces choses du genre :

```
iptables -A FORWARD -i netb -j ACCEPT
iptables -A FORWARD -o netb -j ACCEPT
```

dans le réseau A, et

```
iptables -A FORWARD -i neta -j ACCEPT
iptables -A FORWARD -o neta -j ACCEPT
```

dans le réseau B permettront de laisser passer le trafic dans le tunnel, mais il peut être utile, voire nécessaire, de faire des choses moins permissives.

Note:
La notation de type 172.16.0.0/16 est souvent utilisée pour identifier un réseau. le "/16" indique que les 16 bits les plus lourds (le plus à gauche) sont les seuls bits à considérer pour identifier le réseau.
Autrement dit, ça revient à dire que le masque de sous réseau est 255.255.0.0

Vérifications

Les interfaces

Utilisons iproute2, puisque nous l'avons :

```
gw2:~# ip link list
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,ALLMULTI,UP> mtu 1500 qdisc pfifo_fast qlen 100
link/ether 52:54:05:fc:ad:0c brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,ALLMULTI,UP> mtu 1500 qdisc pfifo_fast qlen 100
link/ether 00:20:af:2f:5d:16 brd ff:ff:ff:ff:ff:ff
25: ppp0: <POINTOPOINT,MULTICAST,NOARP,UP> mtu 1492 qdisc pfifo_fast qlen 3
link/ppp
26: gre0@NONE: <NOARP> mtu 1476 qdisc noop
link/gre 0.0.0.0 brd 0.0.0.0
27: neta@NONE: <POINTOPOINT,NOARP,UP> mtu 1468 qdisc noqueue
link/gre 80.8.147.132 peer 81.248.152.18
```

- 25: ppp0 est le lien PPP vers le réseau du fournisseur d'accès internet,
- 26: gre@NONE est le tunnel lui-même,
- 27: neta@NONE est l'interface virtuelle, qui correspond à l'extrémité du tunnel

Notez le MTU qui diminue à chaque niveau, ce qui est normal. Dans cette situation, nous avons sur Ethernet (niveau 2) :

- Une couche PPP (PPPoE)
    - une couche IP sur ce lien PPP

- un tunnel GRE sur cette couche IP
  - une couche IP dans le tunnel.

Les encapsulations successives entraînent à chaque étape, une diminution de la charge utile des paquets ("payload"), donc le MTU diminue.

Notez bien la particularité de GRE : Ce n'est pas à proprement parler de l'IP sur IP, mais de l'IP dans GRE dans de l'IP. GRE apparaît comme un protocole intermédiaire, nous le reverrons plus pratiquement sur une analyse de trames.

Snif d'un ping

Nous allons faire un petit ping depuis une machine du réseau B (192.168.0.10) vers une machine du réseau A (172.16.252.2).

Nous observons les trames au niveau du routeur B sur l'interface neta (donc au niveau de l'extrémité du tunnel) :

```
Frame 1 (76 bytes on wire, 76 bytes captured)
    Arrival Time: Mar  3, 2004 16:05:17.050838000
    Time delta from previous packet: 0.000000000 seconds
    Time since reference or first frame: 0.000000000 seconds
    Frame Number: 1
    Packet Length: 76 bytes
    Capture Length: 76 bytes
Linux cooked capture
    Packet type: Sent by us (4)
    Link-layer address type: 778
    Link-layer address length: 0
    Source: <MISSING>
    Protocol: IP (0x0800)

# Comme c'est finalement assez logique, nous ne trouvons pas sur cette interface
# de couche  Ethernet.
# Il nous faudrait étudier de façon plus précise le protocole GRE, mais ce n'est
# pas l'objet de ce chapitre.

Internet Protocol, Src Addr: 192.168.0.10, Dst Addr: 172.16.252.2
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 60
    Identification: 0x1b61 (7009)
    Flags: 0x00
        .0.. = Don't fragment: Not set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 127
    Protocol: ICMP (0x01)
    Header checksum: 0x9d0c (correct)
    Source: 192.168.0.10 (192.168.0.10)
    Destination: 172.16.252.2 (172.16.252.2)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x2d5c (correct)
    Identifier: 0x0200
    Sequence number: 0x1e00
    Data (32 bytes)


0000  61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70   abcdefghijklmnop
0010  71 72 73 74 75 76 77 61 62 63 64 65 66 67 68 69   qrstuvwabcdefghi
```

Et la réponse :

```
Frame 2 (76 bytes on wire, 76 bytes captured)
    Arrival Time: Mar  3, 2004 16:05:17.177500000
    Time delta from previous packet: 0.126662000 seconds
    Time since reference or first frame: 0.126662000 seconds
    Frame Number: 2
    Packet Length: 76 bytes
    Capture Length: 76 bytes
Linux cooked capture
    Packet type: Unicast to us (0)
    Link-layer address type: 778
    Link-layer address length: 0
    Source: <MISSING>
    Protocol: IP (0x0800)
Internet Protocol, Src Addr: 172.16.252.2, Dst Addr: 192.168.0.10
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 60
    Identification: 0x067e (1662)
    Flags: 0x00
        .0.. = Don't fragment: Not set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 127
    Protocol: ICMP (0x01)
    Header checksum: 0xb1ef (correct)
    Source: 172.16.252.2 (172.16.252.2)
    Destination: 192.168.0.10 (192.168.0.10)
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0x355c (correct)
    Identifier: 0x0200
    Sequence number: 0x1e00
    Data (32 bytes)


0000  61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70   abcdefghijklmnop
0010  71 72 73 74 75 76 77 61 62 63 64 65 66 67 68 69   qrstuvwabcdefghi
```

Ca marche. Vu au niveau de l'interface neta, tout se passe comme nous avons l'habitude de le voir sur un réseau IP. Notez toutefois que le sniffeur ne reconnaît pas la couche de niveau 2 (ce qui est surligné en bleu pâle)

Conclusions

Mesurez bien la portée de ce que nous venons de faire...

Au travers de l'internet, nous avons créé une liaison spécialisée virtuelle qui permet de relier entre eux deux réseaux IP. Ces deux réseaux sont constitués avec des IP **privées**, et semblent simplement inter-connectés par un routeur. Une adresse IP privée du réseau A dialogue sans problème avec une autre adresse IP privée du réseau B, et réciproquement. Pourtant, le lien entre ces deux réseaux est bel et bien bâti sur l'internet, où ces adresses IP privées sont bannies.

En d'autres termes, Si le réseau B est le réseau de votre lieu de travail et le réseau A celui de votre domicile, vous pouvez par exemple :

- depuis chez vous sous Windows, accéder à votre répertoire partagé sur votre lieu de travail par le voisinage réseau Microsoft,
- en utilisant la connexion du bureau à distance, vous pouvez depuis chez vous travailler sur votre poste de travail professionnel,

- et d'une manière générale, depuis chez vous, vous pouvez faire tout ce que vous faites sur votre lieu de travail.

Bien entendu, vous êtes tout de même limité par le débit de votre connexion. Vous aurez un tuyau d'environ 128 kbps dans le cas le plus courant d'une connexion de type ADSL, rien de comparable, donc, avec un réseau local qui sera au minimum de 10 Mbps.

Mais encore une fois attention !!!

Ce type de tunnel n'est absolument pas sécurisé, vous l'ais-je déjà dit ?

- Les données ne sont pas chiffrées dans le tunnel,
- les deux extrémités du tunnel ne disposent d'aucun processus d'authentification.

Donc, si ce tunnel est en théorie magnifique, en pratique, il l'est beaucoup moins.

Dommage...

Heureusement, d'autres solutions plus sécurisées existent, qui permettent d'aboutir au même résultat sans prendre autant de risques. Ces solutions ne sont pas abordées pour l'instant dans ce chapitre, basées sur le chiffrement et l'authentification.

Le protocole : que disent les RFCs ?

Elles le disent en anglais ici[8]. Désolé, mais personne n'a eu l'idée (même pas moi) de traduire ça en français.

En réalité, pour comprendre comment ça fonctionne, il suffit de comprendre :

- que GRE est considéré comme un protocole de niveau supérieur, qui sera transporté par IP. Pour IP, GRE est donc quelque chose à transporter, au même titre que TCP, UDP, ICMP...
- que GRE va transporter des paquets de niveau 3 (IP, IPX...), ce sera de l'IP le plus souvent.

La capture de trames qui suit montre encore une fois un simple ping, mais observé cette fois-ci sur l'interface ppp0, c'est à dire l'interface qui sert de support au tunnel dans notre exemple. On y observe clairement les deux couches IP l'une dans l'autre :

```
Frame 5 (108 bytes on wire, 108 bytes captured)
    Arrival Time: May 11, 2004 10:09:55.596430000
    Time delta from previous packet: 1.731613000 seconds
    Time since reference or first frame: 1.731613000 seconds
    Frame Number: 5
    Packet Length: 108 bytes
    Capture Length: 108 bytes
Raw packet data
    No link information available
Internet Protocol, Src Addr: 80.8.147.232 (80.8.147.232), Dst Addr: 81.248.157.2 (81.248.157.2)
## ça, c'est la première couche IP, celle qui est transportée par PPPoE sur notre configuration.
## Observez les adresses IP source et destination, ce sont celles de nos passerelles
## sur l'internet.
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 108
    Identification: 0x0000 (0)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 255
    Protocol: GRE (0x2f)
## Vient ensuite GRE sur cette première couche IP
## Notez que pour cette couche IP, GRE est vu comme un protocole de niveau supérieur
## (code de protocole 0x2F) au même titre que TCP (code proto 0x06),
## UDP (code proto 0x11) ou ICMP (code proto 0x01)
    Header checksum: 0xa877 (correct)
    Source: 80.8.147.232 (80.8.147.232)
    Destination: 81.248.157.2 (81.248.157.2)
Generic Routing Encapsulation (IP)
    Flags and version: 0000
        0... .... .... .... = No checksum
        .0.. .... .... .... = No routing
        ..0. .... .... .... = No key
        ...0 .... .... .... = No sequence number
        .... 0... .... .... = No strict source route
        .... .000 .... .... = Recursion control: 0
        .... .... 0000 0... = Flags: 0
        .... .... .... .000 = Version: 0
    Protocol Type: IP (0x0800)
Internet Protocol, Src Addr: 192.168.0.15 (192.168.0.15), Dst Addr: 172.16.254.2 (172.16.254.2)
## Vient enfin la seconde couche IP, encapsurée dans GRE. Observez les adresses IP source
## et destination : Ce sont celles des passerelles dans les deux réseaux privés.
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
```

8 RFC 2784 : http://www.faqs.org/rfcs/rfc2784.html

```
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 84
    Identification: 0x0000 (0)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 63
    Protocol: ICMP (0x01)
    Header checksum: 0xd0de (correct)
    Source: 192.168.0.15 (192.168.0.15)
    Destination: 172.16.254.2 (172.16.254.2)
Internet Control Message Protocol
# Enfin, ICMP (niveau 4), pour le ping.
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xaf0c (correct)
    Identifier: 0xa30d
    Sequence number: 0x0001
    Data (56 bytes)

0000  45 00 00 6c 00 00 40 00 ff 2f a8 77 50 08 93 e8   E..l..@../.wP...
0010  51 f8 9d 02 00 00 08 00 45 00 00 54 00 00 40 00   Q.......E..T..@.
0020  3f 01 d0 de c0 a8 00 0f ac 10 fe 02 08 00 af 0c   ?...............
0030  a3 0d 00 01 53 8a a0 40 be 16 09 00 08 09 0a 0b   ....S..@........
0040  0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b   ................
0050  1c 1d 1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b   .... !"#$%&'()*+
0060  2c 2d 2e 2f 30 31 32 33 34 35 36 37               ,-./01234567
```

Et nous retrouvons la même structure dans la trame de retour :

```
Frame 6 (108 bytes on wire, 108 bytes captured)
    Arrival Time: May 11, 2004 10:09:55.659309000
    Time delta from previous packet: 0.062879000 seconds
    Time since reference or first frame: 1.794492000 seconds
    Frame Number: 6
    Packet Length: 108 bytes
    Capture Length: 108 bytes
Raw packet data
    No link information available
Internet Protocol, Src Addr: 81.248.157.2 (81.248.157.2), Dst Addr: 80.8.147.232 (80.8.147.232)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 108
    Identification: 0x0000 (0)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 251
    Protocol: GRE (0x2f)
    Header checksum: 0xac77 (correct)
    Source: 81.248.157.2 (81.248.157.2)
    Destination: 80.8.147.232 (80.8.147.232)
Generic Routing Encapsulation (IP)
    Flags and version: 0000
        0... .... .... .... = No checksum
        .0.. .... .... .... = No routing
        ..0. .... .... .... = No key
        ...0 .... .... .... = No sequence number
        .... 0... .... .... = No strict source route
        .... .000 .... .... = Recursion control: 0
        .... .... 0000 0... = Flags: 0
        .... .... .... .000 = Version: 0
    Protocol Type: IP (0x0800)
Internet Protocol, Src Addr: 172.16.254.2 (172.16.254.2), Dst Addr: 192.168.0.15 (192.168.0.15)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
```

```
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 84
    Identification: 0xa541 (42305)
    Flags: 0x00
        .0.. = Don't fragment: Not set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 254
    Protocol: ICMP (0x01)
    Header checksum: 0xac9c (correct)
    Source: 172.16.254.2 (172.16.254.2)
    Destination: 192.168.0.15 (192.168.0.15)
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0xb70c (correct)
    Identifier: 0xa30d
    Sequence number: 0x0001
    Data (56 bytes)

0000  45 00 00 6c 00 00 40 00 fb 2f ac 77 51 f8 9d 02   E..l..@../.wQ...
0010  50 08 93 e8 00 00 08 00 45 00 00 54 a5 41 00 00   P.......E..T.A..
0020  fe 01 ac 9c ac 10 fe 02 c0 a8 00 0f 00 00 b7 0c   ................
0030  a3 0d 00 01 53 8a a0 40 be 16 09 00 08 09 0a 0b   ....S..@........
0040  0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b   ................
0050  1c 1d 1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b   .... !"#$%&'()*+
0060  2c 2d 2e 2f 30 31 32 33 34 35 36 37               ,-./01234567
```

Notez tout de même ici la remarquable organisation des couches d'un réseau. En réalité, au sens strict, nous avons un tunnel dans un tunnel, puisque notre tunnel GRE est creusé dans une couche IP elle même "tunnelisée" par PPPoE sur Ethernet du moins, jusqu'à votre modem. Au delà, nous ne savons pas exactement comment ça se passe, ce qui n'a d'ailleurs que peu d'importance, du moment que nous avons une couche IP cohérente et fonctionnelle de bout en bout.

- GRE n'a pas besoin de savoir comment est transportée la couche IP sur laquelle il opère, ici, il fonctionne sur une couche IP portée par PPP,
- PPP n'a pas besoin de savoir ce qu'il y a dans la couche IP qu'il transporte.

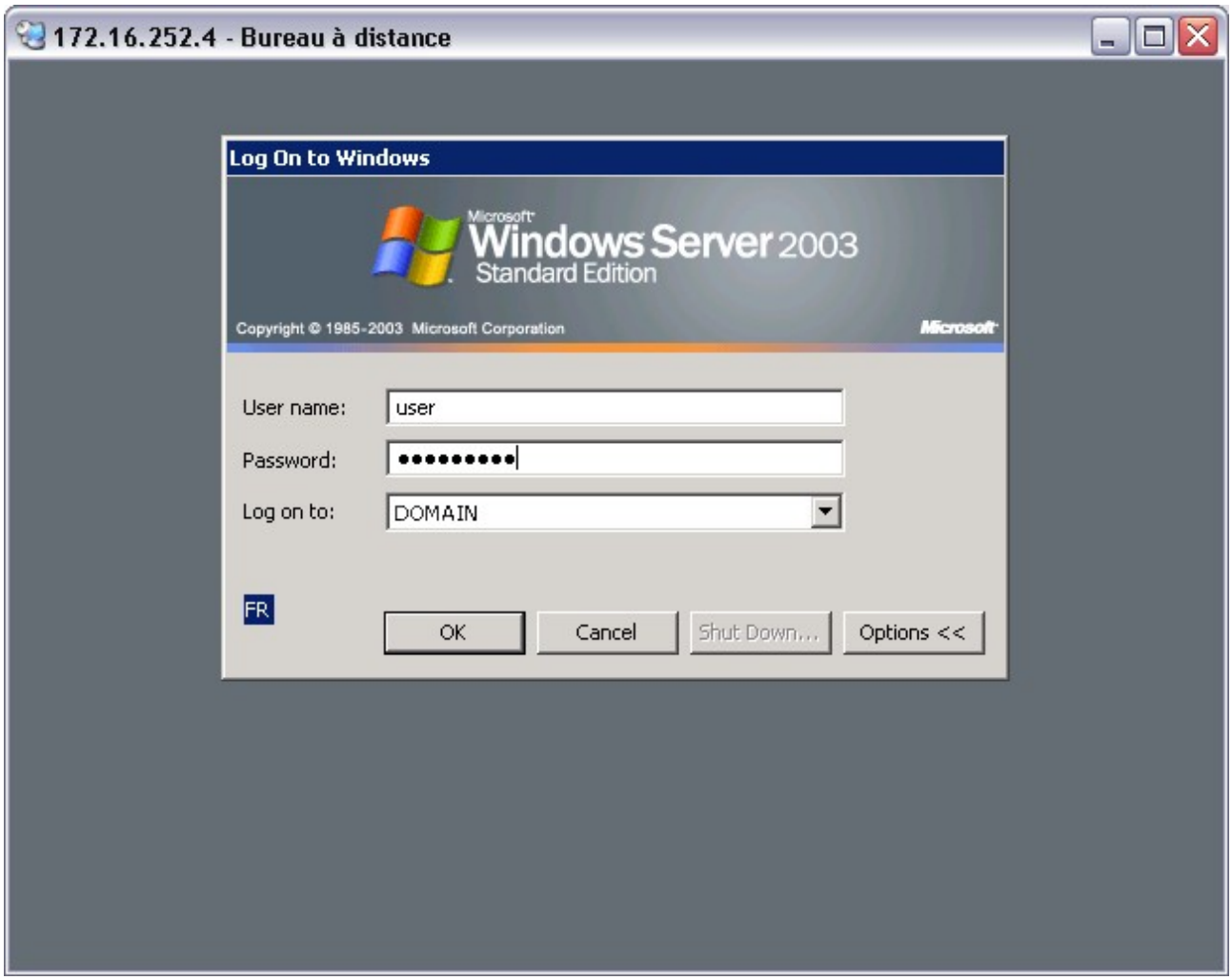
Chacun fait son travail et tout se passe bien. C'est ça la répartition intelligente des tâches.

Utilisation

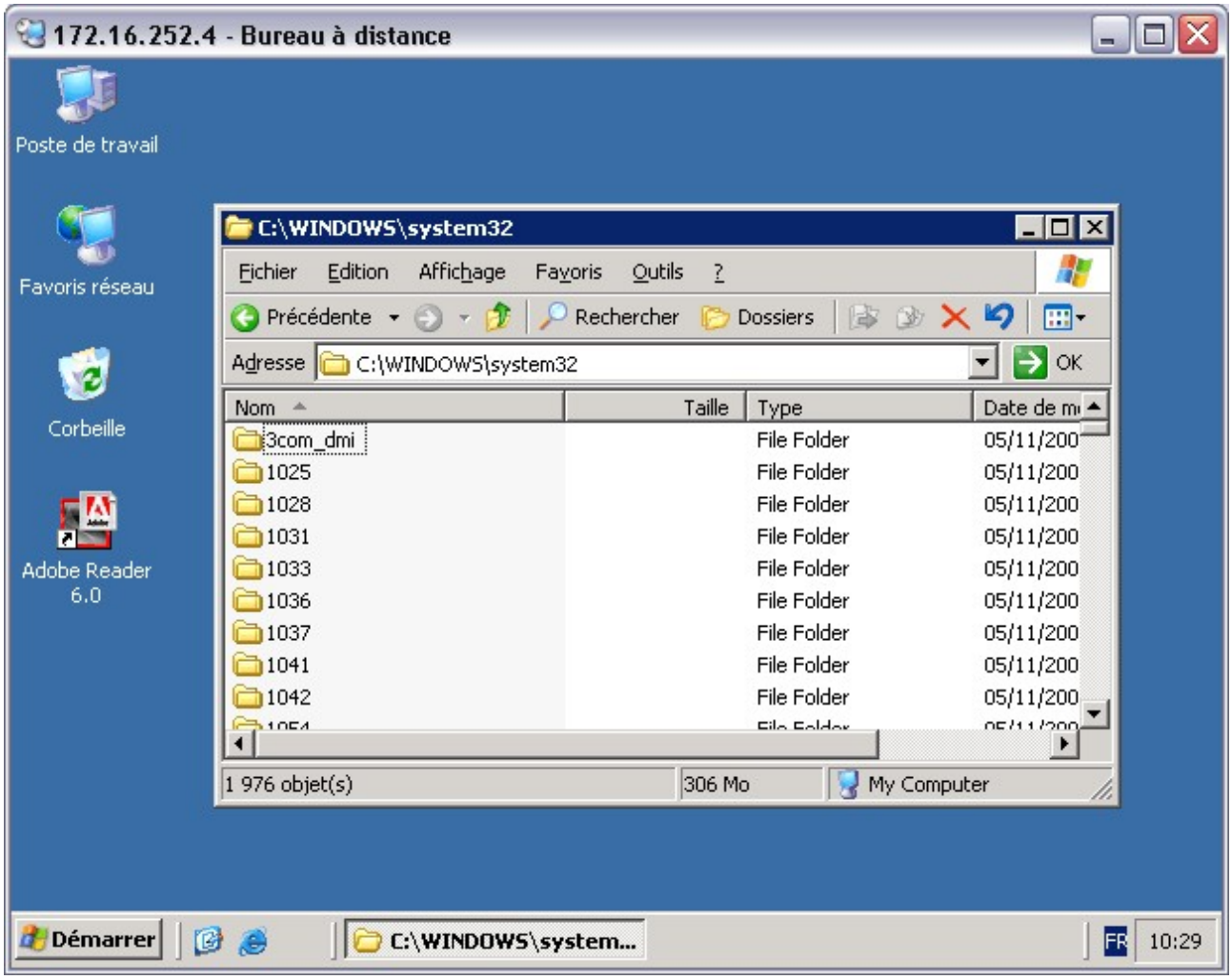
Vous l'avez compris, une fois le tunnel réalisé entre vos deux réseaux, tout se passe comme si ces derniers étaient interconnectés par un routeur, vous êtes chez vous (enfin, tant qu'un intrus ne creuse pas une galerie qui débouche dans votre beau tunnel).

Premier exemple

Dans votre réseau B depuis un poste Windows, vous ouvrez le bureau à distance d'un serveur Windows situé dans le réseau A exactement comme si votre poste de travail était dans le même réseau A. (Pour être tout à fait dans ce cas, il vous faudra éventuellement, régler quelques problèmes de DNS, mais ce n'est pas l'objet de ce chapitre). Dans l'exemple, nous utilisons directement l'adresse IP du serveur :



Et après connexion, nous avons un joli terminal :



Deuxième exemple

Mais ce n'est pas tout, si votre domaine Windows est correctement configuré, que le DNS Active Directory sait résoudre dans les deux réseaux A et B, alors, le tunnel deviendra complètement transparent, vous pourrez, depuis le réseau B ouvrir une session dans le domaine dont le contrôleur se trouve dans le réseau A et votre voisinage réseau vous permettra d'accéder depuis B à des ressources partagées dans le réseau A :

C'est valable, non seulement pour les répertoires partagés, mais aussi pour les imprimantes, bien sûr.

Mais encore une fois, si c'est techniquement possible, c'est tout de même prendre de gros risques de sécurité...

Pour être un peu plus tranquille, il faudrait utiliser un tunnel un peu plus sécurisé, qui chiffre les données et qui assure l'authentification mutuelle pour les deux bouts du tunnel, avec IPSec, par exemple. Mais ça, c'est une autre histoire...

Troisième exemple

Plus techniquement, GRE est souvent utilisé pour faire communiquer deux réseaux IPv6 (adresses IP de 128 bits, soit 16 octets) à travers un réseau IPv4.

IPv6 est le successeur d'IPv4. Beaucoup de problèmes liés à "l'Internet Protocol" actuel seront résolus, principalement la pénurie d'adresses IP et la gestion laborieuse des tables de routage qui s'allongent démesurément avec le fractionnement des classes IP, mais aussi, la sécurité, la qualité de services seront nativement prises en compte. Malheureusement, la mise à niveau d'un réseau planétaire IPv4 en IPv6 va demander beaucoup de temps et d'investissements en matériel. Des solutions de transition seront donc nécessaires.

Limites

Juste un exemple pour montrer au moins qu'un tel tunnel GRE n'offre pas de confidentialité. Nous allons, par l'intermédiaire du "voisinage réseau", utiliser le tunnel pour copier un fichier local sur un hôte distant, à l'autre bout du tunnel.

Ce fichier texte, tout simple, contient le texte : "transfert d'un document par un tunnel GRE".

Le sniffer, mis en service sur l'un des bouts du tunnel, observe ce qu'il passe sur l'interface ppp0. Je ne vous laisse que la trame importante et vous constaterez que les données sont parfaitement lisibles...

```
Frame 48 (175 bytes on wire, 175 bytes captured)
    Arrival Time: May 13, 2004 10:51:47.184526000
    Time delta from previous packet: 0.000812000 seconds
    Time since reference or first frame: 11.459164000 seconds
    Frame Number: 48
    Packet Length: 175 bytes
    Capture Length: 175 bytes
Raw packet data
    No link information available
Internet Protocol, Src Addr: 80.8.147.132 (80.8.147.132), Dst Addr: 81.248.152.18 (81.248.152.18)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 175
    Identification: 0x0000 (0)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 255
    Protocol: GRE (0x2f)
    Header checksum: 0xa5dd (correct)
    Source: 80.8.147.132 (80.8.147.132)
    Destination: 81.248.152.18 (81.248.152.18)
Generic Routing Encapsulation (IP)
    Flags and version: 0000
        0... .... .... .... = No checksum
        .0.. .... .... .... = No routing
        ..0. .... .... .... = No key
        ...0 .... .... .... = No sequence number
        .... 0... .... .... = No strict source route
        .... .000 .... .... = Recursion control: 0
        .... .... 0000 0... = Flags: 0
        .... .... .... .000 = Version: 0
    Protocol Type: IP (0x0800)
Internet Protocol, Src Addr: 192.168.0.10 (192.168.0.10), Dst Addr: 172.16.254.6 (172.16.254.6)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 151
    Identification: 0xc0db (49371)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 127
    Protocol: TCP (0x06)
    Header checksum: 0xcfbb (correct)
    Source: 192.168.0.10 (192.168.0.10)
    Destination: 172.16.254.6 (172.16.254.6)
Transmission Control Protocol, Src Port: 1450 (1450), Dst Port: microsoft-ds (445), Seq: 525021426,
Ack: 4108893321, Len: 111
```

```
    Source port: 1450 (1450)
    Destination port: microsoft-ds (445)
    Sequence number: 525021426
    Next sequence number: 525021537
    Acknowledgement number: 4108893321
    Header length: 20 bytes
    Flags: 0x0018 (PSH, ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 1... = Push: Set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 17304
    Checksum: 0xf21a (correct)
NetBIOS Session Service
    Message Type: Session message
    Length: 107
SMB (Server Message Block Protocol)
    SMB Header
        Server Component: SMB
        Response in: 50
        SMB Command: Write AndX (0x2f)
        NT Status: STATUS_SUCCESS (0x00000000)
        Flags: 0x18
            0... .... = Request/Response: Message is a request to the server
            .0.. .... = Notify: Notify client only on open
            ..0. .... = Oplocks: OpLock not requested/granted
            ...1 .... = Canonicalized Pathnames: Pathnames are canonicalized
            .... 1... = Case Sensitivity: Path names are caseless
            .... ..0. = Receive Buffer Posted: Receive buffer has not been posted
            .... ...0 = Lock and Read: Lock&Read, Write&Unlock are not supported
        Flags2: 0xc807
            1... .... .... .... = Unicode Strings: Strings are Unicode
            .1.. .... .... .... = Error Code Type: Error codes are NT error codes
            ..0. .... .... .... = Execute-only Reads: Don't permit reads if execute-only
            ...0 .... .... .... = Dfs: Don't resolve pathnames with Dfs
            .... 1... .... .... = Extended Security Negotiation: Extended security negotiation is
supported
            .... .... .0.. .... = Long Names Used: Path names in request are not long file names
            .... .... .... .1.. = Security Signatures: Security signatures are supported
            .... .... .... ..1. = Extended Attributes: Extended attributes are supported
            .... .... .... ...1 = Long Names Allowed: Long file names are allowed in the response
        Process ID High: 0
        Signature: EDB2A6ED4322F3CD
        Reserved: 0000
        Tree ID: 32777
        Process ID: 65279
        User ID: 6146
        Multiplex ID: 12417
    Write AndX Request (0x2f)
        Word Count (WCT): 14
        AndXCommand: No further commands (0xff)
        Reserved: 00
        AndXOffset: 57054
        FID: 0x0041
        Offset: 0
        Reserved: FFFFFFFF
        Write Mode: 0x0000
            .... .... .... 0... = Message Start: This is NOT the start of a message (pipe)
            .... .... .... .0.. = Write Raw: DON'T use WriteRawNamedPipe (pipe)
            .... .... .... ..0. = Return Remaining: DON'T return remaining (pipe/dev)
            .... .... .... ...0 = Write Through: Write through not requested
        Remaining: 0
        Data Length High (multiply with 64K): 0
        Data Length Low: 43
        Data Offset: 64
        High Offset: 0
        Byte Count (BCC): 44
        Padding: EE
        File Data: 7472616E7366657274206427756E2064...

0000  45 00 00 af 00 00 40 00 ff 2f a5 dd 50 08 96 3f   E.....@../..P..?
0010  51 f8 9d 02 00 00 08 00 45 00 00 97 c0 db 40 00   Q.......E.....@.
0020  7f 06 cf bb c0 a8 00 0a ac 10 fe 06 05 aa 01 bd   ................
```

```
0030  1f 4b 30 f2 f4 e8 bc 89 50 18 43 98 f2 1a 00 00   .K0.....P.C.....
0040  00 00 00 6b ff 53 4d 42 2f 00 00 00 00 18 07 c8   ...k.SMB/.......
0050  00 00 ed b2 a6 ed 43 22 f3 cd 00 00 09 80 ff fe   ......C"........
0060  02 18 81 30 0e ff 00 de de 41 00 00 00 00 00 ff   ...0.....A......
0070  ff ff ff 00 00 00 00 00 00 2b 00 40 00 00 00 00   .........+.@....
0080  00 2c 00 ee 74 72 61 6e 73 66 65 72 74 20 64 27   .,..transfert d'
0090  75 6e 20 64 6f 63 75 6d 65 6e 74 20 70 61 72 20   un document par
00a0  75 6e 20 74 75 6e 6e 65 6c 20 47 52 45 0d 0a      un tunnel GRE..
```

Au minimum, il faudra donc chiffrer au préalable ses données avant de les faire transiter dans ce tunnel, si l'on ne veut pas qu'un indiscret puisse les lire au passage.

Conclusion

GRE, techniquement est un excellent tunnel, il est possible d'ouvrir depuis un hôte donné autant de tunnels que l'on désire, vers différents réseaux distants. C'est une solution fort souple, malheureusement trop peu sécurisée pour être utilisés sans risques.

IPSec propose d'autres solutions, plus sécurisées, mais difficiles à mettre en oeuvre sur des noyaux Linux 2.4. Ce sera probablement plus simple avec la diffusion des noyaux 2.6, qui intègrent IPSec nativement.