TP réseaux
Translation d'adresse, *firewalls*, zonage

Martin Heusse, Pascal Sicard

1 Avant-propos

Les questions auxquelles il vous est demandé de répondre sont indiquées de cette manière.

Il sera tenu compte de la clarté et de la concision de la rédaction des réponses lors de la correction des comptes-rendus.

Si vous désirez conserver *temporairement* une trace de vos travaux, vous pouvez créer un répertoire personnel sur le disque dur des PCs que vous utilisez. Attention, chaque station dispose de son propre disque. Il n'y a aucune garantie de conservation de vos données sur ces machines.

Il est conseillé de repartir avec des routeurs possédant une configuration minimale. Pour cela on peut faire : `erase startup-config` puis `reload`
A la question "...initial configuration dialog" répondre "n".

2 Introduction

Les manipulations effectuées jusqu'à présent vous ont conduit à mettre en place un réseau local, que vous avez ensuite connecté à un réseau extérieur en utilisant un protocole de niveau 3. Ce TP s'inscrit dans la continuité de ces manipulations, et il vous amènera à limiter l'ouverture du réseau local au monde extérieur.

Ce TP illustre également les notions de tunnel IP sécurisé ou non, et montre comment appliquer des traitements particuliers à des trafics précis, ce qui peut consister en un routage spécifique par exemple (en utilisant les *route-map* des routeurs Cisco).

0. Version 15 mars 2022

3 Translation d'adresse ou NAT (*Network Adresss Translation*)

Il est fréquent de se trouver limité par la plage d'adresses publiques dont on dispose :
— Si l'accès à internet se fait à travers un modem téléphonique, ADSL ou câble (*via* PPP), l'adresse IP de la machine connectée est fournie par le fournisseur d'accès au moment de l'établissement de la connexion. Certains modems peuvent être connectés à plusieurs machines, mais les adresses IP supplémentaires sont généralement payantes.
— Une institution ou une entreprise qui dispose d'un champs d'adressage complet de classe C, par exemple, peut se sentir à l'étroit dans cet espace limité de 254 adresses (environ).

La translation d'adresse permet à un routeur de présenter le trafic relatif au réseau qu'il connecte au reste du monde comme émanant de (et convergeant vers) lui seul (NAT Dynamique). Quand un paquet arrive en provenance du réseau local, il est retransmis vers l'extérieur en changeant dans son entête l'adresse source en celle du routeur. L'opération inverse, c'est à dire la substitution de l'adresse destination, est effectuée pour les paquets circulant dans l'autre sens.

Le traitement à appliquer à chaque paquet incident est déterminé par les adresses IP qu'il porte, mais aussi par les ports du protocole de transport. La translation d'adresse implique donc également souvent une translation de port UDP ou TCP. Le routeur doit maintenir une liste des connexions en cours qui lui permet de déterminer ce qu'il doit faire d'un paquet (translation d'adresse et routage ou bien passage au couches supérieures).

Pour que la translation d'adresse puisse fonctionner, un certain nombre d'opérations annexes doivent être effectuées :
— le checksum des entêtes IP doit être recalculé ;
— le checksum des paquets TCP et UDP doit aussi être recalculé ;
— si un protocole spécifie une adresse dans le champs données, elle doit aussi être modifiée.

Bien sûr il est important que les adresses IP utilisées au sein du réseau local n'existent pas – en étant visibles – ailleurs. Si c'était le cas, les machines qui les porteraient à l'extérieur du réseau local ne seraient pas joignables.

Pourquoi ?

Il existe trois ensembles d'adresses qui sont réservées pour l'usage local :
10.*.*.*
172.16.*.* – 172.31.*.*
192.168.*.*

Les interfaces du « réseau local » utilisé ci-dessous devront donc porter des adresses issues d'un de ces ensembles.

Sur le réseau de la figure 1 les machines C et D forment avec le routeur l'Intranet,

et la machine A représente l' « extérieur ». Attention, utilisez un routeur 800, 2600 ou 4000, le NAT ne fonctionne pas sur les 2500 !

Construire le réseau de la figure 1., configurer les interfaces avec des adresses privées pour l'Intranet et publiques pour l'extérieur.

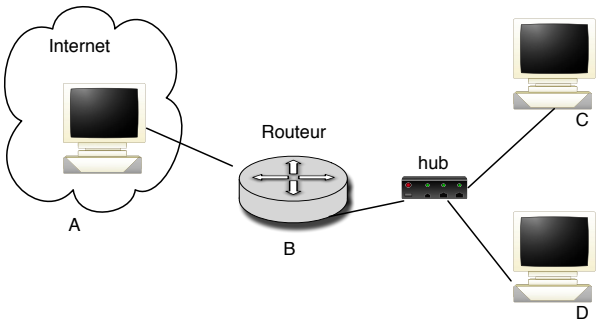

FIGURE 1 – Le réseau

Configurez la table de routage de C et D. On ne touche pas à la table de routage de A, ni à celle de B. Pourquoi ?

3.1 La translation d'adresse sur les routeurs CISCO

Pour que la translation d'adresse marche sur un routeur CISCO, il suffit de désigner les interfaces mise en jeu pour le NAT.
Il faut tout d'abord configurer les deux interfaces utilisées par le NAT en entrée : `inside` (Intranet) et et en sortie : `outside`(externe) :
`ip nat {inside | outside}` suivant l'interface. A faire au niveau de la configuration de l'interface (`Configure terminal` puis `interface`).

Puis on définit l'ensemble des adresses que l'on peut utiliser comme adresse externe. On définit un `pool` d'adresses (sous `configure terminal`) :
`ip nat pool mypool 110.0.0.10 110.0.0.10 prefix-length 8` par exemple spécifie l'utilisation d'une seule adresse externe (110.0.0.10).

Enfin il faut définir l'ensemble des adresses qui vont subir la translation. Pour cela on peut par exemple utiliser une liste d'accès qui autorise toutes les adresses du réseau interne. Exemple (sous `configure terminal`) : `access-list 7 permit 192.168.0.0 0.0.255.255`

Et on spécifie de l'utiliser pour la translation d'adresse (sous `configure terminal`) : `ip nat inside source list 7 pool mypool overload`, où 7 est la liste d'accès définie précédemment. Le mot `overload` signifie que l'on va utiliser la même adresse IP externe pour plusieurs stations internes.

Une alternative à cette ligne de commande est de ne pas utiliser de `pool`, mais de simplement spécifier une interface externe :
`ip nat inside source list 7 interface ethernet 0/0 overload`
Dans ce cas il est inutile de créer un `pool`.

3.2 Ça marche !

Sans NAT, si C « *ping* » A, les ECHO_RESPONSEs de A n'atteignent pas C (pas de route).

Mettre en route la translation d'adresse.

A l'aide de `socklab` ouvrir une connexion TCP depuis une des machines internes vers l'extérieur.

Observez les paquets circulant sur les deux réseaux avec ***Wireshark****, et résumez les opérations effectuées pour la translation d'adresse.*

Pour vous aider regardez la table de translation du NAT par la commande *show ip nat translations*.

À partir de quelle information le routeur identifie-t-il les connexions TCP (ou les flots UDP) de façon à retransmettre les paquets correspondant vers la bonne destination au sein du réseau local ? Expliquez en vous appuyant sur l'observation de la table NAT.

Essayer avec plusieurs connexions TCP simultanées.

Observez les ports utilisés. Il y a t-il toujours aussi une translation de port ? Expliquez en vous appuyant sur l'observation de la table NAT.

Imaginez une expérimentation permettant d'observer une translation de port sur le routeur.

Observez la table Nat lors d'un ping de C vers A.

Quelle information permet de translater correctement les paquets émis et reçus par ***ping*** *? (il n'y a pas de port en ICMP !)*

Comment le routeur peut-il savoir qu'un ECHO_REPONSE portant son adresse comme destination ne lui est en fait pas destiné ?

4 Tunnel

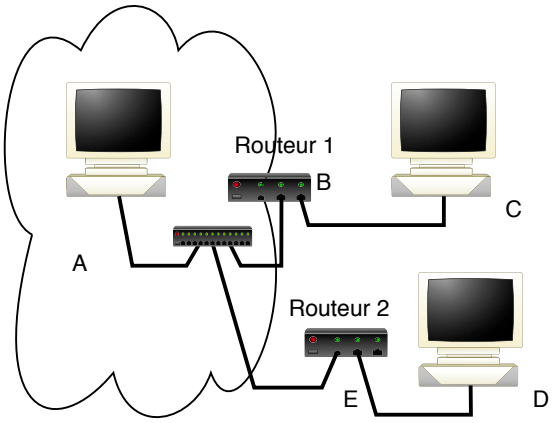

FIGURE 2 – Réseau sur deux sites

Si une partie des stations du réseau dont on a la charge sont situées en dehors du site principal, on se retrouve avec une configuration semblable à celle de la figure 2 : Les routeurs ont des adresses IP publiques sur les interfaces connectées au *hub* qui représente le monde extérieur.

Il peut être « pratique » d'accéder à la station D de manière transparente depuis C, et vice-versa. Pour cela, il ne suffit pas d'ajouter les bonnes entrées dans la table de routage dans le routeur 1 : l'adresse de D est privée, et par conséquent non routée à l'extérieur.

Pour résoudre le problème, il faut établir un tunnel entre le routeur 1 et le routeur 2 (voir figure 3).

4.1 Création d'un tunnel

Rajoutez un routeur à votre montage comme sur la figure 2.

Pour créer un tunnel, il suffit de créer des interfaces virtuelles dites *tunnel* sur les routeurs 1 et 2.
Puis on donne à ces interfaces des adresses IP qui appartiennent au réseau (privé) distant. Il faut ensuite spécifier les adresses IP (publiques) source et destination pour le paquet qui transportera les paquets re-encapsulés par IP à l'entrée du tunnel. Cela se fait en

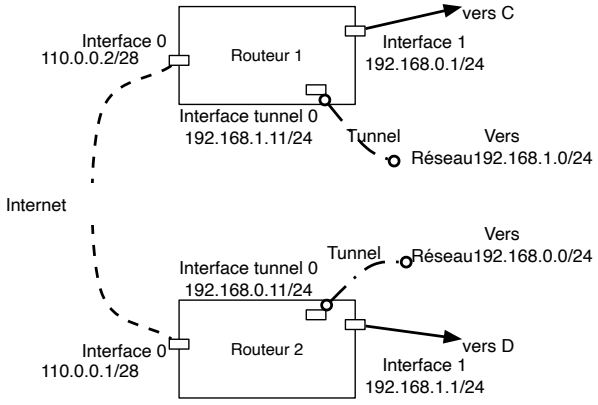

FIGURE 3 – Tunnel

mode de configuration d'interface.
Exemple :

```
interface Tunnel 0
tunnel source 110.0.0.1
tunnel destination 110.0.0.2
ip address 192.168.0.11 255.255.255.0
```

Hop.

Faites en sorte que l'on puisse pinguer D depuis C.

Capturer des paquets en cours de transit dans le tunnel.
Quel est le code du protocole IP pour IP (champs protocol de l'entête) ?

Pourquoi les paquets reçu par le routeur 1 depuis C sont ils redirigés vers l'interface tunnel ?

4.2 Utilisation de demi-ponts

Les machines C et D ne sont pas sur le même réseau (d'un point de vue adressage). Vous avez été obligé de donner une route par défaut à C et D pour que cela fonctionne.

Il est possible de changer l'adressage de C et D afin qu'elles "soient" sur le même réseau. Pour cela il suffit de changer le netmask /24 en /23 (*192.168.0.x/23* et *192.168.1.x/23* appartiennent bien au même réseau.

Changer le netmask sur C et D. Refaites un ping vers D depuis C et observez les paquets sur C.
Quelles différences avec la configuration précédente ? Observez attentivement la requête et la réponse ARP entre C et le routeur 1.

Pour vous aider à comprendre : les routeurs 1 et 2 répondent aux requêtes ARP destinées aux stations situées sur le même sous réseau que l'interface `tunnel` (*proxy-ARP*). On dit que chaque routeur est un demi-pont. Il faut alors bien choisir le plan d'adressage des réseaux de C et D. Les préfixes doivent être "contigus" et l'union donne le préfixe privé qu'on s'est choisi.

4.3 Centralisation du NAT

La configuration actuelle est telle que A et D ne peuvent pas communiquer. Pourquoi ?

Plutôt que de mettre en place de la translation d'adresse sur le routeur 2, il est possible d'utiliser celle en place sur le routeur 1.

Il faut pour cela que tout paquet émis par D à destination de l'extérieur passe d'abord par le tunnel – et subir la translation d'adresse sur le routeur 1 – avant de continuer sa route vers l'extérieur. Il faut donc trouver un moyen pour spécifier sur le routeur 2 un routage particulier des paquets venant du réseau privé.

Comment faire pour que tous les paquets issus du sous-réseau privé de D passent par le tunnel ? (Voir du côté des `route-map`*.)*

ATTENTION, ne pas oublier de spécifier que les paquets issus de l'interface tunnel doivent aussi subir une translation d'adresse (`ip nat inside`).

Exemple de `route-map`

```
! Cette config. prend les paquets qui arrivent sur l'interface 0/1 et
! dont l'adresse commence par 192.168.1.0/24, et les envoie dans le tunnel
interface Tunnel0
 ip address 192.168.0.11 255.255.255.0
 no ip directed-broadcast
 tunnel source 100.0.0.1
 tunnel destination 100.0.0.2
!
interface Ethernet0/0
 ip address 100.0.0.1 255.0.0.0
 no ip directed-broadcast
```

```
!
interface Ethernet0/1
 ip address 192.168.1.1 255.255.255.0
 no ip directed-broadcast
 ip policy route-map intunnel
!
access-list 101 deny   ip host 192.168.1.11 any
access-list 101 permit ip 192.168.1.0 0.0.0.15 any
route-map intunnel permit 10      ! intunnel est le nom de la route-map
match ip address 101
set ip next-hop 192.168.0.1   ! adresse de redirection
```

Les paquets qui passent dans le tunnel sont fragmentés si nécessaire. On peut observer la fragmentation en émettant des paquets « ping » de grande taille à l'aide de l'option -s de ***ping****.*

Le fait de passer dans un tunnel diminue la taille maximale des paquets. Pouvez-vous observez comment se passe cette adaptation dans le cas d'un trafic TCP et UDP ?

Pour cela capturez des paquets TCP de grande taille (émis par exemple par `socklab`). Notez le phénomène observé (ou plutôt le non-phénomène observé...). Pour voir tout ce qui se passe, il faut capturer tous les paquets émis et reçus par l'émetteur d'un paquet de grande taille. La capture doit être faite sur l'émetteur et non dans le tunnel.

Attention, il faut capturer les premiers paquets de grande taille qui passent dans le tunnel. Après c'est trop tard.
Refaites l'expérience avec UDP. Conclusions ?

Remarque : En général dans une telle configuration de tunnel, tous les paquets transitant dans le tunnel sont de plus chiffrés avant d'être envoyés sur Internet (voir manip sur IP Sec plus loin).

5 Mise en œuvre d'un firewall

5.1 Remarques préliminaires

L'utilisation de la translation d'adresse interdit tout établissement de connexion depuis l'extérieur. Par exemple, il n'est pas possible d'avoir un serveur *web* situé à l'intérieur du réseau local. Pour des raisons de sécurité ou pour des raisons pratiques, il est également courant de ne pas le faire tourner sur le routeur lui-même.
Plusieurs solutions sont possibles alors :

— Mettre en place du NAT Static pour certaines machines de l'Intranet.
— Mettre en place du "Port Forwarding" sur le routeur NAT. (voir le cours)
— Disposer d'adresses publiques supplémentaires et construire un deuxième sous-réseau visible de l'extérieur (zone démilitarisée : DMZ). Dans ce TP nous ne mettrons pas en place cette DMZ et conservons un seul routeur filtrant en entrée de l'Intranet.

La règle générale (sécuritaire) pour la mise en œuvre d'un *firewall* est de commencer par bloquer tout, pour ensuite laisser passer exclusivement les paquets identifiés comme acceptables. Cette approche garantit également l'absence de « faille » temporaire au moment où l'on change les règles de filtrage.

5.2 *access-list*

Vous devez déjà connaître l'utilisation des listes de restriction d'accès de CISCO. Mémento sur les listes de restriction d'accès de CISCO :

— La ligne :
`access-list 101 permit tcp 0.0.0.0 255.255.255.255 129.88.38.1 0.0.0.0 eq 80`
autorise le passage des paquets TCP à destination de la station 129.88.38.1 sur le port 80 ; quelque soit l'adresse source et le port source (de l'émetteur). Ici les bits désignés par les masques sont les bits sans importance (à l'opposé de ce qui se passe pour les masques de sous-réseau) ; 0.0.0.0 255.255.255.255 désigne donc effectivement une adresse quelconque. On peut aussi utiliser les mots clés `any host`.

— `access-list 101 deny ip any any` interdit le passage de tous les paquets. Cette ligne termine en général les *access-lists*, afin de bloquer tous les paquets non explicitement décrits plus haut dans la même *access-list*.
Le numéro de de ce type d'*access-list* doit être supérieur à 100.

— Pour mettre en service cette liste d'accès de numéro 101, il faut configurer l'interface sur laquelle le filtrage devra prendre place. Sous `configure interface X` : `ip access-group 101 in`.
Attention le `in` spécifie que le filtrage s'applique aux paquets entrant sur l'interface (depuis le réseau avant la décision de routage) On peut aussi effectuer du filtrage en sortie (vers le réseau, après la décision de routage).

Remarque : Ce type de filtrage (un seul sens) permet dans certains cas de ne pas préciser des règles d'autorisation pour les paquets circulant dans l'autre sens.

Remarque : Le filtrage peut être fait avant ou après le NAT suivant l'interface sur laquelle on le met en place.

5.3 Manipulations

Autoriser à l'aide de liste d'accès sur le routeur 1 :

1. Le trafic depuis la machine C vers un serveur « web » extérieur.
2. Les connexions ssh depuis l'Intranet (machines C et D) vers l'extérieur.
3. Interdire tous le reste.

Comme on ne dispose pas de serveur web, vous pouvez le simuler par une socket à l'aide de *socklab* sur le port de votre choix.
Pour lancer le client ssh on peut taper *ssh guest@host* (mot de passe : guest./).

Donnez les règles de filtrage que vous mettez en place, précisez sur quelle interface et dans quel sens.

Vérifiez le filtrage .

ATTENTION, à ne pas filtrer les paquets circulant dans le tunnel.

En oubliant les contraintes liées au NAT, est il possible d'initier une connexion depuis l'extérieur de l'Intranet avec vos règles ? C'est à dire un utilisateur sur A sur peut il ouvrir une connexion TCP sur C ou D ? Pourquoi ?

Comment filtrer toutes les demandes de connexions TCP depuis l'extérieur quels que soient les numéros de port en jeu ?

6 *Proxy* transparent

Il est possible de diriger – à l'insu des utilisateurs – des paquets vers une *socket* locale au serveur, ou sur une autre machine. Cela permet de forcer par exemple le trafic web à passer par un proxy, sans que l'utilisateur ne puisse s'en apercevoir. La mise en place d'un proxy peut avoir deux raisons. La première est la construction d'un cache local afin de limiter le trafic réseau pour des documents susceptibles d'être consultés de nombreuses fois. La deuxième est le filtrage plus fin du trafic – avec des motivations plus ou moins avouables – pour un protocole donné (le *web*, en général).

Nous supposons que le proxy est situé sur la machine A (Cela pourrait être dans la DMZ). Mettre en place la redirection des paquets sur le routeur 1 vers le proxy quelle que soit l'adresse destination.

Voici un exemple de redirection à l'aide de `route-map`.

```
interface Ethernet0/1
 ip address 192.168.0.254 255.255.255.0
 no ip directed-broadcast
 ip policy route-map proxytrans
! sur l'interface 0/1 les paquets seront rediriger (router) suivant le
```

```
! route-map appele proxytrans

access-list 101 permit ip 192.168.0.0 0.0.0.255 any
route-map proxytrans permit 10
 match ip address 101
! Pour toute les adresses verifiant l'acces list 101
 set ip next-hop 55.1.1.1
! le paquet est envoye à 55.1.1.1
```

Pour tester la redirection, on pourra faire un ping (ou autre à l'aide de *Socklab*) depuis une machine de l'Intranet vers une adresse quelconque et vérifier que les paquets arrivent toujours sur A (adresse Ehernet destination sera celle de A).

Quand ce trafic arrive sur cette station *proxy*, il se présente comme n'étant pas destiné à cette dernière (la destination IP n'est pas l'adresse locale évidemment). Il faudrait donc en plus accepter ces paquets au niveau IP et les rediriger au niveau TCP sur le proxy.

Faire une recherche sur le WEB et expliquez comment cela peut se faire sous free-BSD ou sous Linux (du côté de ipnat et rdr).

7 Séparation des trafics

Le développement de l'utilisation des commutateurs en remplacement des répéteurs contribue grandement à la sécurisation des réseaux, par simple confinement des trafics. L'utilisation de LANs distincts ou de VLANs permet d'obtenir une isolation complète entre différents groupes d'utilisateurs, en forçant les paquets émis d'une entité administrative vers une autre à passer par un routeur – et par conséquent par une liste d'accès ; même si ces paquets circulent sur un câble unique.

Cependant, dans certains cas, il est difficile de se garder des observateurs indiscrets. C'est ce qui se passe pour les réseaux sans fils qui sont par essence indiscrets. Des solutions de bas niveau ont été prévues – comme par exemple sur les réseaux sans fil 802.11b : *Wired Equivalent Privacy* (WEP) – mais elles sont généralement considérées comme insuffisantes.

8 Sécurité par cryptographie

8.1 SSH

`SSH` permet de se « loguer » ou d'exécuter une commande sur une station distante. Par rapport aux commandes classiques comme rlogin ou rsh, `ssh` assure des communications sécurisées (cf. `man ssh` sur un FreeBSD pour plus de détails). Toute une famille de

programmes est disponible avec ssh :

— `scp` qui permet de copier des fichiers vers une station distante ;
— `sftp` : pas la peine de vous faire un dessin ;
— `sshd` : idem.

SSH fait plus que simplement fournir un shell à l'utilisateur. Il peut prendre en charge la translation sécurisée de session X, ou translater un port d'une station à une autre (*port forwarding*).
On peut ainsi accéder à un serveur d'une application tel que FTP, serveur Web, serveur de mail... en utilisant un tunnel *ssh* par exemple pour passer à travers un pare-feu qui n'autoriserait que les connexions *ssh*.

Voici un exemple de l'accès à un serveur *ftp* à travers un tunnel *ssh*. Tunnel entre la machine locale et la machine d'adresse 192.0.0.1 (voir la figure 4) :

```
/usr/bin/ssh -l guest -g -L 1234:192.0.0.1:21 192.0.0.1 -N
```

`guest` est le compte à utiliser sur la machine distante (mot de passe : guest./). Attention on ne peut pas utiliser le compte `root` avec `ssh` (mesure de sécurité).
Un serveur doit donc être à l'écoute sur le port `21` sur la machine `192.0.0.1`

Les paquets envoyés sur le port 1234 de la machine locale arriveront à travers le tunnel ssh sur le port 21 de la machine 192.0.0.1

Donc pour utiliser le tunnel sur la machine locale, ouvrir une connexion vers `127.0.0.1`, et le port `1234`.

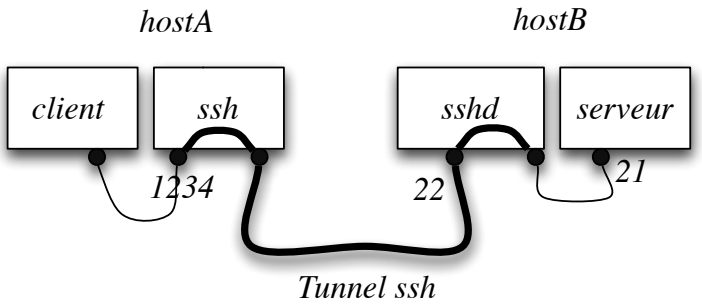

FIGURE 4 – Tunnel ssh

Mettre en place un tunnel SSH entre les machines C et A.
Tester l'utilisation de ce tunnel et la translation de port avec **`socklab`**.
*Envoyer et capturer des paquets (***`Wireshark`***), vérifier qu'ils passent bien à travers le tunnel et que leur contenu est chiffré.*

On peut du coup remarquer que l'on peut "passer" à travers le pare-feu mis en place précédemment quelque soit l'application utilisée.

8.2 SSL

La librairie Ssl donne au programmeur la possibilité de créer des connexions (*sockets*) sécurisées. Du point de vue du programmeur, il faut d'abord créer une *socket* classique, puis l'associer à une entité ssl (cf. `SSL_set_fd()`). Ensuite, la connexion est initiée (`SSL_connect()` et `SSL_accept()`) puis on peut l'utiliser avec `SSL_write()` et `SSL_read()`.

La sécurité offerte par la famille d'outils `OpenSSH/OpenSSL` est tout à fait satisfaisante dans les contextes qui sont les leurs. Cependant, ce type de sécurité fourni au niveau « application » demande à l'utilisateur de savoir ce qu'il fait.
Par exemple `ssh` est systématiquement utilisé pour l'administration à distance, et `ssl` permet de communiquer avec un bon degré de confidentialité avec un serveur SMTP.
Mais pour une sécurisation globale dans un contexte sensible (réseau sans fil, connexion depuis un LAN « étranger ») l'utilisation de tunnels IPSec (également appelés **VPN**) est la solution à privilégier.

8.3 Tunnels IPSec

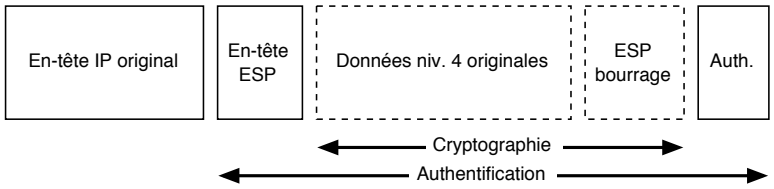

FIGURE 5 – Paquet IPSec (ESP)

IPSec permet d'assurer au niveau IP à la fois le brouillage du contenu des paquets (confidentialité), et leur authentification. L'authentification consiste (simplement ?) à ajouter une signature au paquet. L'utilisation de cette fonctionnalité uniquement est le mode AH de IPSec.
Le mode ESP garantit en plus le chiffrement du trafic, et la forme de paquets envoyés est détaillé en figure 5

IPSec est fréquemment utilisé pour sécurisé un tunnel construit entre deux sites. On se retrouve alors avec des paquets ayant la forme donnée en figure 6. Ainsi il n'est pas possible à un observateur externe de savoir même quelles sont les stations qui s'échangent des paquets à travers le tunnel.

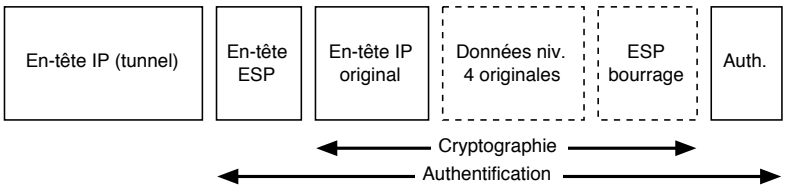

FIGURE 6 – Paquet IPSec/GRE/IP

8.3.1 IPSec sur FreeBSD

Supposons que deux machines A et C possèdent une adresse publique leur permettant un accès à Internet (voir la figure 7). La machine B représente ici Internet. Nous allons créer un **VPN** (Virtual Private Network) entre les deux machines et utiliser IpSec pour chiffrer et/ou authentifier les échanges dans ce VPN.

8.3.2 Création d'un tunnel

Les deux machines A et C vont être connectées virtuellement sur un même réseau local (le VPN). Pour cela nous créons sur chacune des machines une interface virtuelle. Les adresses IP associées à ces deux interfaces virtuelles seront celle d'un même réseau (par exemple une adresse privée : *192.168.0.0/24*). Il faut ensuite créer un tunnel à travers Internet permettant de relier physiquement ces deux interfaces virtuelles. Les extrémités de ce tunnel sont les interfaces physiques des machines.

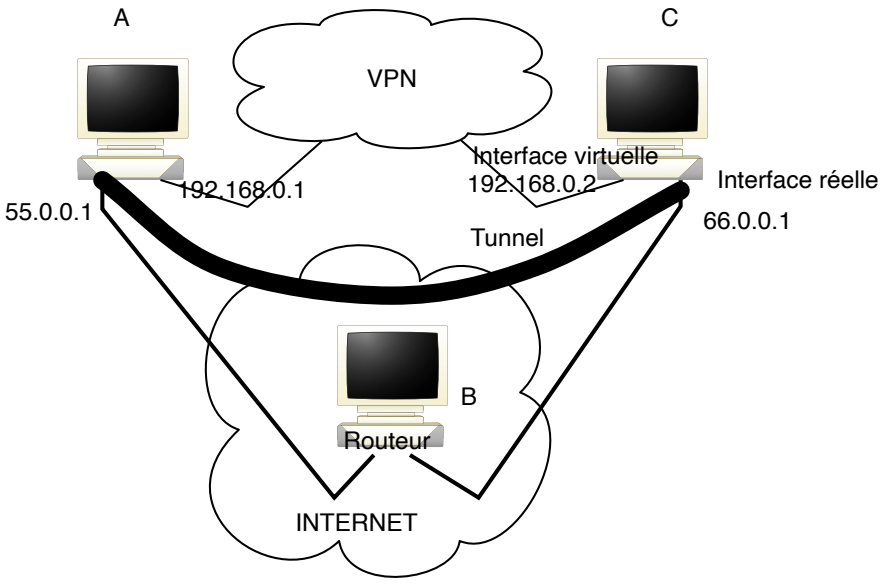

FIGURE 7 – Tunnel de bout en bout

Les paquets routés vers ces interfaces virtuelles seront automatiquement ré-encapsuler à l'aide d'une deuxième entête IP portant les adresses publiques des bouts du tunnels. Ils sont ensuite émis sur les interfaces portants les adresses des bouts de tunnels (forcément publiques). A l'arrivée à l'autre bout du tunnel, la deuxième entête IP est enlevée, le processus de routage reprend sur la deuxième entête.

Remarque : On pourrait de la même façon mettre en place un accès sécurisé à l'ensemble d'un Intranet en réalisant un tunnel entre une machine extérieure et le routeur d'entrée de l'Intranet (comme cela est fait à l'UFR-IM2AG). Il est aussi possible aussi de créer un tunnel entre les routeurs d'entrée de deux Intranet (comme dans l'expérimentation précédente avec les demi-ponts).

Comment créer un tunnel sous FreeBSD ? Exemple de création d'une interface virtuelle associée à un tunnel (voir figure 7) :

```
ifconfig gif0 create
ifconfig gif0 tunnel 55.0.0.1 66.0.0.1
ifconfig gif0 192.168.0.2/24 192.168.0.1
```

Créer un tunnel entre les machines A et C en utilisant une adresse publique pour les bouts du tunnel. Observer la table de routage des machines. Capturer des paquets transitant dans le tunnel.

8.3.3 Sécurisation du tunnel

Les machines A et C sont reliés au même réseau, il faut maintenant le "privatisé", c'est à dire faire en sorte que les données qui transite à travers le tunnel et donc Internet soit chiffrées.
On peut aussi mettre en place des mécanismes d'authentification pour accéder au VPN.

Sous Free-BSD, la première chose à faire est de préciser au noyau quels sont les paquets qui doivent être encodés. La table SPD (*Security Policy Database*) contient ces informations, et la commande `setkey` permet de la gérer. `setkey` permet aussi d'associer manuellement des clés d'encodage aux communications spécifiées, les informations correspondantes apparaissant dans la table SAD (*Security Association Database*).

`setkey -F`
`setkey -FP`
vident les tables SPD ET SAD.

On peut ensuite les remplir en leur passant un fichier de commandes (`setkey -f` *file*).

Par exemple, la ligne :

```
spdadd 192.168.0.2/32 192.168.0.1/32  any
              -P out ipsec esp/tunnel/55.0.0.1-66.0.0.1/require;
```

spécifie d'encoder tous les trafics transitant par le tunnel dont les adresses "internes" sont 192.168.0.1 et 192.168.0.2, et dont les adresses "externes" sont 55.0.0.1 et 66.0.0.1. Il faut donner une directive semblable afin de désigner quels paquets incidents doivent être décodés.

Remarque on pourrait ici aussi utiliser le mode "Transport" de IPSEC, il suffit alors de donner :

```
spdadd AdrPriveeSource/32 AdrPriveeDest/32 any
     -P out ipsec esp/transport/AdrPubSource-AdrPubDest/require;
```

La ligne `add src dst esp spi -E ealgo clé`
permet ensuite d'associer une clé à l'encodage du trafic visé. `ealgo` est l'algorithme d'encryption utilisé, `clé` est la clé correspondante. (cf. `man setkey`.) `Spi` est un nombre supérieur à 255 – par exemple `0x10003`.

Voici un exemple de configuration sur A pour déchiffrer les paquets arrivant du tunnel :

```
spdadd 192.168.0.1/24 192.168.0.2/24  any
                -P in ipsec esp/tunnel/66.0.0.1-55.0.0.1/require;
add 66.0.0.1 55.0.0.1 esp 0x10003 -E des-cbc "12345678";
```

Le pendant sur la machine C pour chiffrer les paquets envoyés dans le tunnel :

```
spdadd 192.168.0.1/24 192.168.0.2/24  any
                -P out ipsec esp/tunnel/66.0.0.1-55.0.0.1/require;
add 66.0.0.1 55.0.0.1 esp 0x10003 -E des-cbc "12345678";
```

On peut utiliser aussi un algorithme d'authentification par exemple sur A :

```
spdadd 192.168.0.0/24 192.168.0.0/24 any -P in ipsec
    ah/transport/66.0.0.1-55.0.0.1/require ;
add 66.0.0.1 55.0.0.1 ah 123456 -m any -A hmac-md5  "0123456789123456" ;
```

La longueur de la clé de l'algorithme d'authentification `hmac-md5` doit être exactement de 16 octets.

Sur C :

```
spdadd 192.168.0.0/24 192.168.0.0/24 any -P out ipsec
    ah/transport/66.0.0.1-55.0.0.1/require ;
add 66.0.0.1 55.0.0.1 ah 123456 -m any -A hmac-md5  "0123456789123456" ;
```

On peut chiffrer et authentifier en même temps, par exemple :

```
spdadd 192.168.0.0/24 192.168.0.0/24 any -P out ipsec
    ah/transport/66.0.0.1-55.0.0.1/require  esp/tunnel/66.0.0.1-55.0.0.1/require;
add 66.0.0.1 55.0.0.1 ah 123456 -m transport -A hmac-md5  "0123456789123456" ;
add 66.0.0.1 55.0.0.1 esp 0x10003  -m tunnel -E des-cbc "12345678" ;
```

Remarque FreeBSD utilise une encapsulation IP dans IP plutôt que GRE (Generic Routine Encapsulation).

Chiffrer et/ou authentifier les trafics qui transitent par le tunnel précédemment créé. Capturer des paquets dans les deux cas : authentification et chiffrement (AH et ESP). Commentaires. À quoi sert l'index ***`spi`*** *?*

ATTENTION : il faut capturer les paquets sur l'interface physique pour observer le chiffrement.

Un fichier est à disposition sur le Moodle.

Est il possible d'utiliser un tel tunnel en utilisant à l'une des extrémités une adresse privée translatée par un routeur intermédiaire. Pourquoi ?

Remarque : Ces tunnels IPSec utilisent une clé statique. Ce qui pose des problèmes de sécurité sur le long terme. Il faudrait avoir recours à un protocole (et à des démons) d'échange de clé IKE (Internet Key Echange).

Références

[1] D. Reed : IP Filter - TCP/IP Firewall/NAT Software.
http ://coombs.anu.edu.au/∼avalon/ip-filter.html.

[2] L. Toutain : *Réseaux locaux et internet.* Hermès, 1999.