Évaluation en environnement réel de la mise en œuvre des services de sécurité dans les architectures typiques - Aspects liés à ICMP -

Logiciels-
Réseaux

Jean-Jacques Puig
Maryline Laurent-Maknavicius

04004-LOR
2004

Avril 2004

EVALUATION EN ENVIRONNEMENT REEL DE LA MISE EN ŒUVRE DES SERVICES DE SECURITE DANS LES ARCHITECTURES TYPIQUES - ASPECTS LIES A ICMP

---oOo---

Abstract : This document reports ICMP testing aiming to study whether ICMP interactions with IPsec mi result in possible network security degradations.

Key words: IPsec, ICMP

Résumé : Ce rapport rend compte de manipulations effectuées avec ICMP dans le but de déterminer si interactions de ce protocoles avec IPsec peuvent provoquer une détérioration réelle de la sécurité du réseau

Mots-clés : IPsec, ICMP

---oOo—

Jean-Jacques Puig
Doctorant
Institut National des Télécommunications
Département LOR
9 rue Charles Fourier 91011 Evry cedex
E-mail: Jean-Jacques.Puig@int-evry.fr

Maryline Laurent-Maknavicius
Maître de conférernce
Institut National des Télécommunications
Département LOR
9 rue Charles Fourier 91011 Evry cedex
E-mail: Maryline.Maknavicius@int-evry.fr

Avril 2004

Évaluation en environnement réel de la mise en œuvre des services de sécurité dans les architectures typiques - Aspects liés à ICMP -

Jean-Jacques Puig

Jean-Jacques.Puig@int-evry.fr

INT

Maryline Laurent-Maknavicius

Maryline.Maknavicius@int-evry.fr

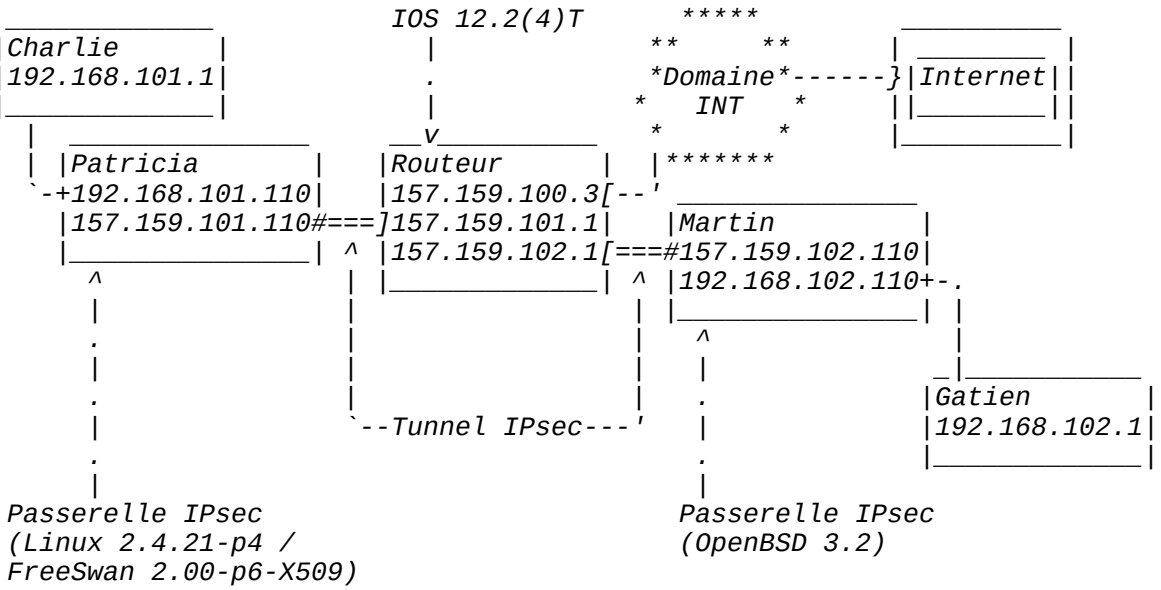
INT

Avril 2004

SOMMAIRE

1. Introduction

La ``connectivité TCP/IP'' (les accès réseaux sont parfois qualifiés ainsi) implique l'utilisation de nombreux protocoles connexes, souvent mal compris, et par conséquent mal maîtrisés.
Ce rapport rend compte de manipulations effectuées avec [ARP] et [ICMP] dans le but de déterminer si les interactions de ces protocoles avec IPsec peuvent provoquer une détérioration réelle de la sécurité du réseau.
Dans un premier document, nous avons mené une analyse d'ARP. Nous traitons dans celui-ci des spécificités de ICMP.
Côté ``intrus'', les manipulations présentées dans ce document ont été réalisées sous [Linux] (Kernel 2.4). Côté ``entreprise'', nous avons utilisé deux passerelles de sécurité, une sous Linux (Kernel 2.4.21-pre4 avec [FreeSwan] 2.00-pre6-X509) et une sous [OpenBSD] 3.2.
Le schéma ci-dessous donne la description du montage :



1. Retour aux Sources...

1.1. Remarques préliminaires

Les codes ICMP existants sont référencés sur le site de [IANA]. Dans le cadre de nos manipulations, nous ne nous sommes intéressés qu'aux codes utilisés en IPv4 et standardisés par un RFC, ce qui nous donne la liste suivante :

Type	Nom	Référence
0	_Echo_Reply	[RFC792]
3	_Destination_Unreachable	[RFC792]
4	_Source_Quench	[RFC792]
5	_Redirect	[RFC792]
6	_Alternate_Host_Address	[JBP]
8	_Echo	[RFC792]
9	_Router_Advertisement	[RFC1256]
10	_Router_Solicitation	[RFC1256]
11	_Time_Exceeded	[RFC792]

```
12      _Parameter_Problem                          [RFC792]
13      _Timestamp                                  [RFC792]
14      _Timestamp_Reply                            [RFC792]
15      _Information_Request                        [RFC792]
16      _Information_Reply                          [RFC792]
17      _Address_Mask_Request                       [RFC950]
18      _Address_Mask_Reply                         [RFC950]
30      _Traceroute                                 [RFC1393]
31      _Datagram_Conversion_Error                  [RFC1475]
32      _Mobile_Host_Redirect                       [David Johnson]
35      _Mobile_Registration_Request                [Bill Simpson]
36      _Mobile_Registration_Reply                  [Bill Simpson]
37      _Domain_Name_Request                        [Simpson]
38      _Domain_Name_Reply                          [Simpson]
39      _SKIP                                       [Markson]
40      _Photuris                                   [RFC2521]
```

Avec les codes correspondants :

```
Type    Nom                                     Référence
----    -------------------------                ---------
 0      Echo_Reply                                [RFC792]
           Codes
            0  No Code

 3      Destination_Unreachable                   [RFC792]
           Codes
            0  _Net_Unreachable
            1  _Host_Unreachable
            2  _Protocol_Unreachable
            3  _Port_Unreachable
            4  _Fragmentation_Needed_and_Don't_Fragment_was_Set
            5  _Source_Route_Failed
            6  _Destination_Network_Unknown
            7  _Destination_Host_Unknown
            8  _Source_Host_Isolated
            9  _Communication_with_Destination_Network_is_
                Administratively_Prohibited
           10  _Communication_with_Destination_Host_is_
                Administratively_Prohibited
           11  _Destination_Network_Unreachable_for_Type_of_Service
           12  _Destination_Host_Unreachable_for_Type_of_Service
           13  _Communication_Administratively_Prohibited [RFC1812]
           14  _Host_Precedence_Violation                 [RFC1812]
           15  _Precedence_cutoff_in_effect               [RFC1812]

 4      Source_Quench                             [RFC792]
           Codes
            0  No Code

 5      Redirect                                  [RFC792]
           Codes
            0  _Redirect_Datagram_for_the_Network_(or_subnet)
            1  _Redirect_Datagram_for_the_Host
            2  _Redirect_Datagram_for_the_Type_of_Service_and_Network
            3  _Redirect_Datagram_for_the_Type_of_Service_and_Host

 6      Alternate_Host_Address                       [JBP]
           Codes
            0  _Alternate_Address_for_Host

 8      Echo                                      [RFC792]
           Codes
            0  No Code
```

```
 9      Router_Advertisement                        [RFC1256]
           Codes
            0  _Normal_router_advertisement
           16  _Does_not_route_common_traffic      [RFC2002]

10      Router_Selection                            [RFC1256]
           Codes
            0  No Code

11      Time_Exceeded                                [RFC792]
           Codes
            0  _Time_to_Live_exceeded_in_Transit
            1  _Fragment_Reassembly_Time_Exceeded

12      Parameter_Problem                            [RFC792]
           Codes
            0  _Pointer_indicates_the_error
            1  _Missing_a_Required_Option          [RFC1108]
            2  _Bad_Length

13      Timestamp                                    [RFC792]
           Codes
            0  No Code

14      Timestamp_Reply                              [RFC792]
           Codes
            0  No Code

15      Information_Request                          [RFC792]
           Codes
            0  No Code

16      Information_Reply                            [RFC792]
           Codes
            0  No Code

17      Address_Mask_Request                         [RFC950]
           Codes
            0  No Code

18      Address_Mask_Reply                           [RFC950]
           Codes
            0  No Code

30      Traceroute                                   [RFC1393]

31      Datagram_Conversion_Error                    [RFC1475]

32      Mobile_Host_Redirect                    [David Johnson]

35      Mobile_Registration_Request              [Bill Simpson]

36      Mobile_Registration_Reply                [Bill Simpson]

39      SKIP                                          [Markson]

40      Photuris                                     [RFC2521]
           Codes
            0  _Bad_SPI
            1  _Authentication_Failed
            2  _Decompression_Failed
            3  _Decryption_Failed
            4  _Need_Authentication
            5  _Need_Authorization
```

Par ailleurs, les outils que nous avons utilisés (voir plus loin) ne sont pas capables de générer tous ces types. Cela n'a cependant pas une grande conséquence, compte tenu des types ICMP supportés par les noyaux des OS utilisés ; les deux extraits suivants présentent les définitions des types ICMP pour Linux et OpenBSD :

Extrait de `include/linux/icmp.h` *(Linux) :*

```
#define ICMP_ECHOREPLY          0       /* Echo Reply                   */
#define ICMP_DEST_UNREACH       3       /* Destination Unreachable      */
#define ICMP_SOURCE_QUENCH      4       /* Source Quench                */
#define ICMP_REDIRECT           5       /* Redirect (change route)      */
#define ICMP_ECHO               8       /* Echo Request                 */
#define ICMP_TIME_EXCEEDED      11      /* Time Exceeded                */
#define ICMP_PARAMETERPROB      12      /* Parameter Problem            */
#define ICMP_TIMESTAMP          13      /* Timestamp Request            */
#define ICMP_TIMESTAMPREPLY     14      /* Timestamp Reply              */
#define ICMP_INFO_REQUEST       15      /* Information Request          */
#define ICMP_INFO_REPLY         16      /* Information Reply            */
#define ICMP_ADDRESS            17      /* Address Mask Request         */
#define ICMP_ADDRESSREPLY       18      /* Address Mask Reply           */
```

Extrait de `sys/netinet/ip_icmp.h` *(OpenBSD) :*

```
#define ICMP_ECHOREPLY          0       /* echo reply */
#define ICMP_UNREACH            3       /* dest unreachable, codes: */
#define ICMP_SOURCEQUENCH       4       /* packet lost, slow down */
#define ICMP_REDIRECT           5       /* shorter route, codes: */
#define ICMP_ALTHOSTADDR        6       /* alternate host address */
#define ICMP_ECHO               8       /* echo service */
#define ICMP_ROUTERADVERT       9       /* router advertisement */
#define ICMP_ROUTERSOLICIT      10      /* router solicitation */
#define ICMP_TIMXCEED           11      /* time exceeded, code: */
#define ICMP_PARAMPROB          12      /* ip header bad */
#define ICMP_TSTAMP             13      /* timestamp request */
#define ICMP_TSTAMPREPLY        14      /* timestamp reply */
#define ICMP_IREQ               15      /* information request */
#define ICMP_IREQREPLY          16      /* information reply */
#define ICMP_MASKREQ            17      /* address mask request */
#define ICMP_MASKREPLY          18      /* address mask reply */
#define ICMP_TRACEROUTE         30      /* traceroute */
#define ICMP_DATACONVERR        31      /* data conversion error */
#define ICMP_MOBILE_REDIRECT    32      /* mobile host redirect */
#define ICMP_IPV6_WHEREAREYOU   33      /* IPv6 where-are-you */
#define ICMP_IPV6_IAMHERE       34      /* IPv6 i-am-here */
#define ICMP_MOBILE_REGREQUEST  35      /* mobile registration req */
#define ICMP_MOBILE_REGREPLY    36      /* mobile registration reply */
#define ICMP_SKIP               39      /* SKIP */
#define ICMP_PHOTURIS           40      /* Photuris */
```

Les définitions de OpenBSD sont très complètes, puisqu'elles incluent les affectations de IANA pour les types ICMP pour lesquels il n'y a pas encore de RFC. Les distributions *BSD constituent des systèmes de choix pour l'expérimentation de nouveaux protocoles réseaux (du moins est-ce la réputation qu'elles se donnent), ce qui justifie la mention de messages encore expérimentaux (par exemple pour la mobilité). La présence de définitions aussi complètes facilite les nouveaux développements au détriment de la traçabilité des types pour lesquels un traitement est réellement effectué. Le code de sys/netinet/ip_icmp.c est, fort heureusement, très clair sur ce point :

```
/*
 * No kernel processing for the following;
 * just fall through to send to raw listener.
 */
case ICMP_ECHOREPLY:
case ICMP_ROUTERADVERT:
case ICMP_ROUTERSOLICIT:
case ICMP_TSTAMPREPLY:
case ICMP_IREQREPLY:
case ICMP_MASKREPLY:
case ICMP_TRACEROUTE:
case ICMP_DATACONVERR:
case ICMP_MOBILE_REDIRECT:
case ICMP_IPV6_WHEREAREYOU:
case ICMP_IPV6_IAMHERE:
case ICMP_MOBILE_REGREQUEST:
case ICMP_MOBILE_REGREPLY:
case ICMP_PHOTURIS:
...
```

Bien souvent, la conception et la mise en oeuvre de jeux de tests sur un protocole est une tâche difficile. En particulier quand le protocole se compose de nombreux messages et peut faire intervenir une relation entre plus de deux machines (comme cela est le cas pour de nombreux messages d'erreurs de ICMP). Nous avons donc décidé de pousser plus loin l'analyse des sources afin de connaître précisément les traitements opérés par IPsec sur ICMP et d'identifier les messages ICMP réellement traités par les noyaux.
Par soucis d'exhaustivité, nous avons analysé l'ensemble des symboles présents dans les sources de ces deux noyaux ainsi que dans les patchs utilisés sur la maquette, afin de nous assurer qu'il n'y a aucune définition ``à la sauvage'' qui pourrait être liée à ICMP. Les commandes suivantes nous ont permis de construire deux dictionnaires des définitions liées à ICMP rencontrées dans les sources :

```
Cthulhu:~# fgrep --recursive ICMP \
    /usr/src/freeswan-2.00-rc2 \
    /usr/src/linux-2.4.21-pre5 \
    /usr/src/x509-1.1.6-freeswan-2.00-pre6 \
    | tr [:cntrl:] '\n' \
    | tr [:blank:] '\n' \
    | tr [:space:] '\n' \
    | tr '_' ' ' \
    | tr [:punct:] '\n' \
    | tr ' ' '_' \
    | grep ICMP \
    | sort -b -u \
    > Linux.symboles
```

Ce dictionnaire contient un seul type supplémentaire de message ICMP pour IPv4 : ADDRESS_MASK (i.e. messages `_Address_Mask_Request` et `_Address_Mask_Reply`). Après recherche dans les sources, il ne s'agit pas d'une définition, mais d'une discussion sous forme de commentaires sur les raisons qui ont incité les développeurs à retirer le support de ce message dans le noyau (conformément à §3.2.2.9 de [Hosts-Rqts]).

```
Cthulhu:~# fgrep --recursive ICMP \
    /usr/src/OpenBSD \
    | tr [:cntrl:] '\n' \
    | tr [:blank:] '\n' \
    | tr [:space:] '\n' \
    | tr '_' ' ' \
    | tr [:punct:] '\n' \
```

```
    | tr ' ' '_' \
    | grep ICMP \
    | sort -b -u \
    > OpenBSD.symboles
```

Ce dictionnaire ne contient pas de définitions supplémentaires de types ICMP pour IPv4 (ce qui semble normal, puisque `ip_icmp.h` est déjà très complet).
Voyons maintenant plus en détail la gestion de ICMP dans le cadre des traitements opérés par IPsec sur ces deux systèmes. Pour cela, nous avons été amenés à considérer les résultats de la commande suivante, appliquée successivement sur les différents répertoires des sources concernées :

```
Cthulhu:~# fgrep --recursive --ignore-case \
    --files-with-matches --no-messages \
    ICMP <répertoire inspecté>
```

Pour des raisons évidentes de clarté, nous ne faisons ici état que de notre analyse de ces concordances dans les sources :

1.2. IPsec & Linux

X509-FreeSwan : Le patch X509 pour FreeSwan ne contient AUCUN traitement sur ICMP.

FreeSwan (KLIPS) : FreeSwan n'opère en réalité que très peu de traitements liés à ICMP ; ces opérations sont concentrées dans le fichier `ipsec_tunnel.c`, qui décrit l'implémentation du mode tunnel. Quatre comportements sont liés à ICMP :

1) Affichage détaillé des caractéristiques des paquets ICMP si IPsec est exécuté en mode DEBUG : adresse source, checksum, code et type du message.

2) Renvoi d'un message `_Time_Exceeded / _Time_to_Live_exceeded_in_Transit` à l'émetteur d'un paquet si le TTL est nul au moment où le paquet devrait être acheminé dans le tunnel (code repris de `ip_forward.c`).

3) Lorsque les caractéristiques d'un paquet ne correspondent pas à celles spécifiées lors de la construction de l'association de sécurité (via les ``traffic selectors''), le paquet peut être rejeté (configuration dans `ipsec.conf`) au lieu d'être simplement détruit. En réalité, lorsqu'un paquet est soumis à une analyse afin de déterminer s'il convient de l'acheminer dans un tunnel, une association de sécurité est recherchée grâce aux caractéristiques du paquet. L'association de sécurité ainsi déterminée peut être associée à un tunnel ou à des politiques ``DROP'' ou ``REJECT'' (dans ces deux derniers cas, on parle de ``Shunt SA''). Cette dernière politique (action de rejet) se caractérise alors par l'émission d'un message `_Destination_Unreachable / _Communication_Administratively_Prohibited` vers la source du paquet fautif.

4) Lorsqu'un paquet devrait être acheminé dans le tunnel (i.e : son acheminement est ``politiquement'' correct) mais que sa taille (`IP_Total_Length`) s'avère trop importante (i.e : supérieure au MTU du tunnel) et que l'option `IP_Don't_Fragment` est active pour ce paquet, un message `_Destination_Unreachable / _Fragmentation_Needed_and_Don't_Fragment_was_Set` est renvoyé à la source d'un paquet. Cette option est active par défaut, mais peut être désactivée via `sysctl` (variable

`ipsec_icmp`). Suivant la configuration, le tunnel prendra ou non l'initiative d'opérer malgré tout la fragmentation, passant outre le bit `IP_Don't_Fragment`. Les paquets ICMP ne sont pas concernés par ce traitement ([ICMP] précise qu'il est interdit d'envoyer un message ICMP au sujet d'un message ICMP).

Le traitement des paquets en mode tunnel pourrait **à l'avenir** intégrer deux fonctionnalités liées à ICMP :

- La prise en compte des messages `_Destination_Unreachable / _Fragmentation_Needed_and_Don't_Fragment_was_Set` émis par les routeurs intermédiaires pour redimensionner les interfaces du tunnel.

- Rejet des paquets fragmentés arrivant depuis le tunnel (les paquets doivent être réassemblés **avant** traitement par IPsec, et refragmentés **ensuite**).

FreeSwan (Pluto)

`pluto` est le démon IKE de FreeSwan. Le seul traitement apporté spécifiquement par `pluto` lors de la réception d'**une erreur ICMP** est l'affichage - en mode debug - du type et du code ICMP concernés, tout en rappelant que le message ICMP n'est pas authentifié. La gestion du protocole ICMP lui-même est laissée au noyau, et `pluto` reçoit juste une notification d'erreur via une file appropriée.
Les futurs développements prévus pour pluto (fichier `TODO`) envisagent la gestion du type `_Destination_Unreachable` comme indice d'une déconnexion probable des télétravailleurs et de mener plus loin l'investigation de façon fiable et authentifiée afin de rompre les associations de sécurité correspondantes. Cette pratique (une forme de ``Dead Peer Discovery'') n'est ni triviale, ni standardisée dans l'immédiat.

Linux

Cette partie clos notre étude des sources liées à l'utilisation d'IPsec sous Linux, en présentant les mécanismes appliqués par défaut par le noyau, lorsque IPsec n'assume pas lui-même le traitement d'un message ICMP. Comme précédemment, **cette étude est restreinte au seul contexte d'IPv4**.

Nous avons choisi de la scinder en cinq parties :

1. ICMP comme stimulus du système,

2. ICMP comme réaction du système,

3. Acheminement de ICMP,

4. Contrôle utilisateur sur les traitements liés à ICMP,

5. Statistiques sur ICMP.

Acheminement de ICMP

L'acheminement de ICMP est remis en question lorsque le système opère des traitements particuliers sur le réseau. Cela signifie en particulier que son comportement dépasse le simple cadre du routage.

Un premier exemple est le module de commutation de trames (`Frame_Diverter`). Dotés de ce module, le noyau peut se comporter en commutateur, et dans ce cas l'administrateur doit spécifier si les trames contenant des paquets ICMP sont aussi prises en charge lors de la commutation (cela affecte en particulier les notions de broadcast).
Par ailleurs, la translation d'adresses constitue aussi une contrainte d'acheminement. Plusieurs modules du noyau peuvent assurer la translation d'adresse, en se basant sur des techniques différentes. Un de ces modules n'assure que la traduction des messages ICMP de type `_Destination_Unreachable`, `_Time_Exceeded` et `_Parameter_Problem`. Un autre module, intégré au suivi des connections, est capable d'en effectuer la gestion complète. Le problème est cependant relativement subtil, dans la mesure où un paquet ICMP arrivant peut être lié à un paquet qui a lui-même été traduit. Bien sûr, les messages `_Redirect` ne sont pas acheminés.
Le tunneling est aussi une pratique qui implique un traitement spécifique de ICMP. Lorsqu'une extrémité du tunnel reçoit une message d'erreur, il conviendrait de l'acheminer à l'émetteur concerné. Or, trop souvent (Linux est peut être même la seule exception), les informations contenues dans les messages sont limitées au minimum autorisé dans [ICMP]. Pour activer la prise en charge de l'acheminement de ICMP dans un tunnel IP/IP ou GRE/IP, il est nécessaire de définir une variable dans les sources (`I_WISH_WORLD_WERE_PERFECT`), et tout dépend encore de la quantité d'informations qui seront incorporées dans les messages d'erreurs retournés par les intermédiaires !
Enfin, lorsque le message ICMP est traité localement, les différentes couches concernées en effectuent certes le traitement (voir partie suivante), mais le paquet est aussi acheminé vers l'application, qui peut en prendre connaissance via une file d'erreurs appropriée. D'autres systèmes d'acheminement vers l'espace utilisateur existent aussi, tels que les `socket_netlink`.

ICMP comme stimulus du système

Dans notre lecture des sources du noyau, nous avons rencontré un particularisme qui mérite d'être mentionné ici. Ce particularisme concerne une certaine catégorie de cartes réseaux (à base de chipset ``Typhoon'') qui seraient apparemment capables de mettre l'ordinateur sous tension lors de la réception d'un paquet _Echo. Les solutions habituelles pour ce type de mécanismes sont les ``Magic Packets'', l'observation de l'état du lien, et l'écoute des requêtes ARP.
Lors de la réception d'un message ICMP sur l'hôte, ce message est tout d'abord pris en charge par le moteur de filtrage du noyau, ``NetFilter''.
Suivant sa configuration, NetFilter opère plusieurs types de traitements sur les paquets :

- Le module `IP_NF_MATCH_UNCLEAN` permet de repérer des paquets ICMP dont les attributs sont anormaux. L'administrateur peut configurer le noyau pour déclencher des actions comme la journalisation ou le rejet lors de l'occurence de tels paquets. Les caractéristiques du paquet son représentées sous la forme d'un tristate {Valide, Invalide, Indéterminé}. Un traitement particulier est effectué vis à vis du message `_Parameter_Problem`, pour déterminer l'origine du problème à partir des informations retournées.

- Le module `IP_NF_CONNTRACK` est en charge du suivi des flux de paquets. En particulier, il anticipe sur les propriétés des paquets à venir. Par exemple, il peut refuser l'acheminement d'un

message _Echo_Reply si aucun message _Echo aux caractéristiques symétriques n'a été émis. Les réactions déclenchées par l'arrivée d'un message imprévu sont configurables par l'administrateur (acheminement, rejet, journalisation...). Dans le cas des messages d'erreurs, les informations retournées sont parfois insuffisantes pour pouvoir être exploitées.

- Le moteur de suivi de connexion utilise évidemment un système d'identification des paquets. Un tel système est aussi à la disposition de l'administrateur, qui peut donc définir des actions en fonction de l'occurence de certains messages ICMP, caractérisés par leur type et leur code. Cela est particulièrement intéressant pour journaliser des messages ICMP obsolètes et identifier ainsi des routeurs mal configurés ou encore pour détecter les prémices d'attaques. Soulignons que le masque de filtrage est inversible.

- La cible ``LOG'' de NetFilter permet la journalisation des évènements. Quand ces évènements sont des messages ou des erreurs ICMP, une mise en forme spécifique est effectuée de façon à faciliter la compréhension des données du paquet.

- Les comportements sus-mentionnés de NetFilter, accessibles via iptables, sont partiellement utilisables via d'anciennes interfaces comme ipfw et ipchains.

De là, le contenu des trames Ethernet est passé à la couche supérieure, à moins que le noyau n'ait été configuré en relayeur de trames ICMP (Option DIVERT_PROTO_ICMP), auquel cas les messages sont acheminés sans plus de formalités. Jusqu'ici, les statistiques (par exemple, le nombre de paquets perdus) des interfaces n'ont pas été mises à jours.

Si le message ICMP est à destination du système local (cette notion de destination peut notamment englober le broadcast), alors le module principal de gestion de ICMP intervient (lors de l'appel au protocole de transport depuis la couche réseau). Ce module principal opère tous les traitements classiques exigés par les normes pour les routeurs ou les hôtes, suivant la configuration sélectionnée par l'administrateur. Le module prend en charge :

- Les questions _Echo, _Timestamp, en y fournissant une réponse appropriée.

- Les réponses _Address_Mask_Reply, en en vérifiant la cohérence. Tout autre type de réponse est acheminé vers le programme propriétaire du socket à l'origine de la question.

- Les erreurs _Destination_Unreachable, _Source_Quench, _Time_Exceeded, _Parameter_Problem avec les codes associés, en opérant les conversions nécessaires pour les acheminer vers les applications mises en causes.

- Les messages _Redirect, en mettant les routes à jour.

Si le protocole de transport concerné par une erreur ICMP est UDP ou TCP, une remise en forme de l'erreur est effectuée pour la file d'erreurs liée au socket. Plus précisément, l'erreur ICMP est traduite en type d'erreur système. Dans le cas particulier de TCP, des erreurs _Destination_Unreachable / _Fragmentation_Needed_and_Don't_Fragment_was_Set peuvent survenir. Ces erreurs déclenchent le mécanisme de détermination du PMTU, et de redimensionnement des segments TCP le

cas échéant. Le même type d'opération se manifeste pour les encapsulations IP/IP ou GRE/IP, afin de procéder au redimensionnement des interfaces du tunnel.
Dans le contexte de l'utilisation des `sockets_raw`, un filtre ICMP peut être défini. Par ailleurs, lors de la réception de messages d'erreur ICMP, un traitement simple est effectué de façon à connecter ces évènements sur la file d'erreur accessible au niveau du `socket_raw`.
Un cas particulier de message ICMP orienté ``service'' est directement pris en charge par le noyau : il s'agit des erreurs qui rendent compte de l'inaccessibilité des RPCs. Le noyau sert alors d'intermédiaire pour la gestion des erreurs.

Stimuli provoquant l'émission de messages ICMP

Voyons maintenant dans quelles situations le système émet des paquets ICMP :

- Un contexte d'émission un peu particulier est celui de ``CLIP'' en ATM. Si, sur ATM, un paquet IP doit être acheminé vers un circuit virtuel (VC) par le système, et que ce dernier ne trouve par dans la table de correspondance ATM/ARP de circuit virtuel correspondant à la destination, il peut émettre une message `_Destination_Unreachable` / `_Host_Unreachable`.

- Dans le cadre d'un comportement consistant au niveau IP, le système peut se comporter en tant que routeur filtrant. Dans ce cas, il analyse des tuples de la forme `<src prefix, src interface, tos, dst prefix>`. Il peut alors prendre la décision (administrative) de rejeter un paquet, par l'émission de messages `_Destination_Unreachable` / `_Host_Unreachable` ou `_Destination_Unreachable` / `_Communication_Administratively_Prohibited`.

- La couche de filtrage de paquets (NetFilter) assure aussi un certain nombre d'opérations pouvant provoquer l'émission de paquets. En particulier, il est possible de rejeter un paquet et d'émettre les messages suivant : `_Destination_Unreachable` / `_Net_Unreachable`, `_Destination_Unreachable` / `_Host_Unreachable`, `_Destination_Unreachable` / `_Protocol_Unreachable`, `_Destination_Unreachable` / `_Port_Unreachable`, `_Echo_Reply`, `_Destination_Unreachable` / `_Communication_with_Destination_Network_is_Administratively_Prohibited` et `_Destination_Unreachable` / `_Communication_with_Destination_Host_is_Administratively_Prohibited`.

- L'ancien module de filtrage, encore présent dans le noyau pour des raisons de compatibilité, peut rejeter les paquets via un message `_Destination_Unreachable` / `_Port_Unreachable`.

- Le module ``miroir'' de NetFilter permet d'effectuer un effet de miroir sur les paquets qui parviennent sur une interface, i.e. de les renvoyer. Comme le ``hook'' qui déclenche cet action se situe en amont de la couche de routage, il est nécessaire de décrémenter le TTL des paquets in-situ, et donc d'émettre un message `_Time_Exceeded` / `_Time_to_Live_exceeded_in_Transit` le cas échéant.

- Avant d'entrer dans la couche réseau, la validité des champs du paquet est vérifiée. En particulier, les options de IP sont analysées. Si une incohérence apparaît, un message est renvoyé vers la source : `_Parameter_Problem`.

- Le point d'entrée dans la couche réseau n'envoie de messages ICMP que lorsque le protocole de niveau supérieur, s'il existe, ne peut être atteint. Cela ne concerne bien sûr que les paquets à destination du système local. Dans ce cas, les messages ne sont jamais envoyés si l'adresse source est un broadcast, et leur type est `_Destination_Unreachable / _Protocol_Unreachable`.

- Si le paquet est à destination du système local ou si des traitements de type détection d'intrusion, proxy ou autres sont menés dessus, un ré-assemblage est nécessaire. L'émission d'un message `_Time_Exceeded / _Fragment_Reassembly_Time_Exceeded` vient sanctionner une arrivée trop lente des fragments.

- La couche de routage effectue très peu d'opérations donnant lieu à l'émission d'un paquet ICMP. En particulier, la gestion du TTL ne s'effectue pas à cette endroit. En revanche, si une meilleure route pour les paquets est détectée sur l'interface par laquelle ils sont arrivés, la couche émet un message de redirection (`_Redirect / _Redirect_Datagram_for_the_Host`). Par ailleurs, quand la destination ne peut être atteinte, des messages `_Destination_Unreachable` avec l'un des codes `_Host_Unreachable`, `_Net_Unreachable` ou `_Communication_Administratively_Prohibited` sont émis (des `_Host_Unreachable` sont aussi émis si l'interface de sortie a son lien à l'état bas).

- Une fois routé, le paquet est acheminé par le module ip_forward.c. Si l'option de `Source_Routing` est active sur le paquet, une erreur de route peut subvenir, auquel cas un message `_Destination_Unreachable / _Source_Route_Failed` est émis vers la source. C'est aussi au niveau de ce module que la durée de vie du paquet est réduite, avec éventuellement un message `_Time_Exceeded / _Time_to_Live_exceeded_in_Transit` renvoyé à l'émetteur. Enfin, une vérification potentiellement redondante avec le traitement présenté ci-après est effectuée sur le MTU.

- Le point de sortie de la couche réseau peut être amené à émettre des messages `_Destination_Unreachable / _Fragmentation_Needed_and_Don't_Fragment_was_Set` vers la source si le MTU de l'interface de sortie est trop petit et si le bit `Don't_Fragment` était actif.

- Si le paquet était à destination des protocoles de tunneling (IP/IP ou GRE/IP), plusieurs situations peuvent provoquer l'émission d'un message. Tout d'abord, si les adresses mentionnées dans le paquet ne correspondent pas à un tunnel explicitement configuré sur le système, le paquet est rejeté avec un `_Destination_Unreachable / _Protocol_Unreachable`. Ensuite, dans le sens de l'encapsulation, si l'option `Don't_Fragment` du paquet est active et que le MTU de l'interface de sortie est trop important pour permettre l'encapsulation des paquets dans le tunnel, un message `_Destination_Unreachable / _Fragmentation_Needed_and_Don't_Fragment_was_Set` est renvoyé à la source. Enfin, suivant la quantité d'information disponible, ces modules sont capables de gérer l'acheminement d'erreurs ICMP (cf. plus haut).

- Si le paquet était à destination du protocole UDP (couche transport), l'absence d'application en écoute sur le port destination provoque l'émission d'un message `_Destination_Unreachable / _Port_Unreachable` vers la source.

Contrôle utilisateur sur les traitements liés à ICMP

Ce contrôle est triple : via la configuration de la compilation, ou depuis l'espace utilisateur lors de l'exécution via sysctl et procfs.
Lors de la configuration des sources, les éléments suivants sont proposés en option :

- `CONFIG_IP_NF_MATCH_UNCLEAN`

Cette option de NetFilter active le repérage de paquets étranges ou invalides, par exemple des paquets ICMP dont les codes et types sont incohérents.

- `CONFIG_IP_NF_TARGET_REJECT`

Cette option de NetFilter permet l'émission d'une erreur ICMP lorsqu'un paquet correspondant à des critères définis par l'utilisateur est reçu.

- `CONFIG_IP_NF_TARGET_TCPMSS`

Cette option est directement liée à l'absence de message ICMP. Plus précisément, elle permet d'altérer la taille des segments TCP de façon à ne pas provoquer de fragmentation sur un chemin. Comme certains FAI et administrateurs appliquent des politiques de sécurité aveugles sur leurs équipements et empêchent l'émission ``naturelle'' de messages ICMP `_Fragmentation_Needed_and_Don't_Fragment_was_Set`, une solution devait être trouvée pour permettre à l'utilisateur de spécifier lui-même la taille des segments.

- `CONFIG_ATM_CLIP_NO_ICMP`

Cette option pour ATM concerne l'activation ou la désactivation de l'émission de messages `_Host_Unreachable` lorsqu'un circuit virtuel ne peut être trouvé pour la destination dans la table de correspondance ATM/ARP (cf. plus haut).

Les deux autres méthodes de configuration, procfs et sysctl, ont de nombreuses interconnections et permettent de modifier le comportement du noyau lors de l'exécution. Elles affectent les variables suivantes du noyau :

- `icmp_ignore_bogus_error_responses` (booléen, défaut=`false`) : Cette option permet d'ignorer les réponses émises par une certaine catégorie de routeurs défectueux. En temps normal, ces trames erronées sont signalées dans le système d'audit. Cette option permet d'éviter la journalisation de cet évènement.

- `accept_redirects` (booléen, défaut=`false` si le routage est actif, `true` sinon) : Cette option précise si une interface accepte les messages de redirection (_Redirect).

- `secure_redirects` (booléen, défaut=`true`) : Cette option précise si l'hôte ne doit accepter que les messages de redirection émis par ses passerelles.

- `icmp_echo_ignore_all` (booléen, défaut= `true`) : Cette option précise si l'hôte doit ignorer les messages ICMP de demande d'écho (Echo).

- `icmp_echo_ignore_broadcasts` (booléen, défaut= `false`) : Cette option précise si l'hôte doit ignorer les messages ICMP de demande d'écho envoyés vers une adresse de broadcast.

- `send_redirects` (booléen, défaut= `true`) : Cette option précise si l'hôte peut prendre l'initiative d'émettre des messages ICMP de redirection (Redirect). Cela n'est possible que si l'hôte fait du routage (i.e : /proc/sys/net/ipv4/ip_forward == 1).

Les options suivantes permettent de contrôler les flux de messages ICMP, afin d'éviter des dénis de services. Aucune limite n'est cependant imposée pour les types ICMP inconnus, ou pour l'interface de ``loopback''. Le flux du couple `_Destination_Unreachable` / `_Fragmentation_Needed_and_Don't Fragment_was_Set` n'est pas non plus contrôlé afin de faciliter la mise en oeuvre du ``Path MTU Discovery''.

- `error_cost, error_burst` (entiers, défaut=`100, 500` respectivement) : Ces deux paramètres permettent de contrôler le débit de messages `_Destination_Unreachable` émis par la couche de routage, ainsi que d'établir une limite au-delà de laquelle les messages ICMP sont jetés. Conjointement, des entrées au sujet de ces messages sont écrites dans le journal du noyau, selon les mêmes modalités de fréquence et de limite.

- `icmp_destunreach_rate, icmp_echoreply_rate, icmp_paramprob_rate, icmp_timeexeed_rate` (entiers, défaut=`100` Hertz) : Ces options sont encore accessibles via procfs pour contrôler les flux d'émissions des messages `_Destination_Unreachable, _Echo_Reply, _Parameter_Problem` et `_Time_Exceeded`. Cependant, il est conseillé désormais d'utiliser les options ci-dessous.

- `icmp_ratelimit` (entier, défaut=`100` Hertz) : Cette option permet de fixer le débit maximal d'émission des messages dont les types sont masqués par l'option suivante.

- `icmp_ratemask` (entier, défaut=`6168`) : Il s'agit d'un masque de bits permettant de spécifier les types ICMP sur lesquels s'appliquent la limitation de débit mentionnée ci-dessus. En voici la signification bit à bit, depuis le moins significatif :

0 Echo Reply
3 Destination Unreachable (par défaut)
4 Source Quench (par défaut)
5 Redirect
8 Echo Request
B Time Exceeded (par défaut)
C Parameter Problem (par défaut)
D Timestamp Request
E Timestamp Reply
F Info Request
G Info Reply
H Address Mask Request
I Address Mask Reply

Statistiques sur ICMP.

La MIB SNMP contient les données suivantes relatives à ICMP, dont l'interprétation est donnée par [MIB-TCP/IP]. L'installation d'un agent SNMP est nécessaire pour accéder à ces données :

- IcmpInMsgs
- IcmpInErrors

- IcmpInDestUnreachs
- IcmpInTimeExcds
- IcmpInParmProbs
- IcmpInSrcQuenchs
- IcmpInRedirects
- IcmpInEchos
- IcmpInEchoReps
- IcmpInTimestamps
- IcmpInTimestampReps
- IcmpInAddrMasks
- IcmpInAddrMaskReps
- IcmpOutMsgs
- IcmpOutErrors
- IcmpOutDestUnreachs
- IcmpOutTimeExcds
- IcmpOutParmProbs
- IcmpOutSrcQuenchs
- IcmpOutRedirects
- IcmpOutEchos
- IcmpOutEchoReps
- IcmpOutTimestamps
- IcmpOutTimestampReps
- IcmpOutAddrMasks
- IcmpOutAddrMaskReps
- OutOfWindowIcmps
- LockDroppedIcmps

D'autres informations sont cependant disponibles : la date de la dernière erreur ICMP repérée dans un tunnel IP/IP, les tuples `<code, type, id>` utilisés par le système de suivi des connexions, etc. Par ailleurs, certains fichiers sources utilisent des variables préprocesseur pour compiler l'affichage d'informations de débogage relatives à ICMP (`source, destination, code, type,` et les autres informations utiles suivant le type de message).

1.3. IPsec & *OpenBSD*

OpenBSD (IPsec)

L'implémentation d'IPsec dans OpenBSD est native, ce qui rend les sources plus modulaires. En particulier, l'implémentation d'IPsec en mode tunnel utilise réellement l'implémentation IP/IP officielle du noyau (alors que ce n'est pas le cas pour FreeSwan).
Pour mener une comparaison correcte, nous avons donc analysé les parties IPsec et IP/IP du noyau OpenBSD.
Le source de l'encapsulation IP/IP n'effectue aucun traitement relatif à ICMP, bien qu'il soit précisé dedans sous la forme de commentaires que cela serait préférable :-).
Le source de l'implémentation IPsec mentionne un seul traitement de ICMP, lors de la réception de paquets. Plus précisément, ce traitement concerne la lecture du MTU des messages `_Destination_Unreachable` / `_Fragmentation_Needed_and_Don't_Fragment_was_Set` renvoyés par des routeurs de façon à déterminer le MTU du chemin (``path MTU discovery'').

OpenBSD (isakmpd)

isakmpd est le démon [IKE] de OpenBSD (ses sources sont accessibles dans l'archive des sources du ``userland'' de OpenBSD : src.tar.gz). Ce programme ne contient absolument aucune référence à ICMP, ce qui est un peu surprenant : on se serait attendu au moins à un traitement d'erreurs spécifique sur réception des messages ICMP. Cela signifie que le traitement déjà opéré par le noyau n'est pas approfondi par isakmpd lors de l'occurence de tels évènements.

OpenBSD (Core ICMP support)

Comme précédemment dans le cas de Linux, nous avons poursuivi notre étude par l'analyse des mécanismes classiques du noyau touchant à ICMP.

Niveau interfaces virtuelles

Le module assurant la fonction de ponts entre interfaces peut décider d'émettre des messages `_Destination_Unreachable` / `_Fragmentation_Needed_and_Don't_Fragment_was_Set` lorsque le MTU d'un interface ne permet pas l'acheminement du paquet sans fragmentation et que le bit `Don't_Fragment` est activé sur le paquet.
Le module de gestion des lignes séries peut filtrer ICMP de façon à économiser de la bande passante.
Le module d'émission radio-amateur opère le même type de manipulations.

Niveau filtrage

PacketFilter effectue les opérations de filtrage sur les paquets. Ces opérations comportent en particulier le suivi de connexions et la translation d'adresse. Dans le cadre du suivi de connexion, PacketFilter corrèle deux à deux les réponses ICMP à des requêtes ICMP et les erreurs ICMP à des paquets UDP ou TCP émis dans le sens contraire. Par ailleurs, PacketFilter pouvant effectuer le routage et l'acheminement du paquet, il est amené à émettre des erreurs `_Destination_Unreachable` / `_Fragmentation_Needed_and_Don't_Fragment_was_Set` quand la situation le requiert.

Niveau IP

Le module en charge de la réception des paquets émet des messages ICMP appropriées dans les cas suivants :

- Echec du routage de source.
- Erreur de paramètre IP.
- Hôte injoignable.
- Redirection si une meilleure route est connue.
- Mort du paquet (expiration du TTL).

- Taille du paquet trop importante pour l'interface de sortie, et fragmentation interdite.

- Dépassement du temps autorisé pour le réassemblage des fragments.

- Si la mémoire vient à se faire insuffisante pour assurer la réception du paquet, le système émet un message `_Source_Quench`.

Niveau ICMP

Comme cela a été mentionné lors de l'étude du noyau Linux, les sources relatifs à ICMP décrivent en fait peu de traitements sur les messages reçus. En particulier, les messages d'erreurs sont uniquement observés pour des fins de statistiques, puis acheminés vers le protocol concerné par une file d'erreurs.
Les requêtes ICMP directement traitées par ce module sont donc `_Echo, _Timestamp et _Address_Mask_Request`. Ces requêtes déclenchent les réponses correspondantes, selon la configuration spécifiée via sysctl (voir plus bas).
Le seul message pris en compte en plus des requêtes précédentes est la redirection (`_Redirect`). Auquel cas, une route est effectivement ajoutée, avec une durée de vie associée.
Les statistiques suivantes sont gérées par l'automate du protocole, et stockées dans des entiers longs non signés :

- Le nombre de messages d'erreurs renvoyés.

- Le nombre de paquets erronés pour lesquels la quantité d'information disponible était trop faible pour former une réponse.

- Le nombre de paquets erronés de type ICMP pour lesquels il est évidemment interdit d'émettre quelque réponse que ce soit.

- Le nombre de codes ICMP erronés reçus.

- Le nombre de messages ICMP trop courts reçus.

- Le nombre de messages pour lesquels la somme de contrôle était fausse.

- Le nombre de messages pour lesquels la longueur était fausse.

- Le nombre de réponses renvoyées.

- Le nombre de messages envoyés aux broadcast et rejetés.

- Le nombre de paquets émis, par code.

- Le nombre de paquets reçus, par code.

Via sysctl, il est possible d'ajuster les comportements suivants :

- L'émission de réponses à destination du broadcast ou d'adresses multicast.

- L'émission de réponses aux requêtes _Address_Mask_Request.

- L'émission de réponses aux requêtes _Timestamp.

- Le débit des erreurs renvoyées.

- La prise en compte des messages de redirection émis par les routeurs.

- La durée de vie des routes ajoutées suite à la réception d'un message de redirection.

Niveau sockets / transport

Les socket_raw BSD ont une caractéristique particulière qui provoque l'émission d'un message _Destination_Unreachable / _Protocol_Unreachable lorsqu'un paquet reçu ne peut être aligné correctement sur la structure logique.
De façon tout à fait classique, la couche UDP émet un message _Destination_Unreachable / _Port_Unreachable lorsque le port vers lequel est adressée l'unité de protocole de transport est fermé.
L'implémentation de TCP opère les traitements relatifs au ``PMTU discovery'' lorsque cela s'avère nécessaire (cf. implémentation Linux plus haut)
Afin de simplifier les manipulations qui seront menées par la suite, nous avons précédemment mené l'analyse des sources des noyaux jusqu'à obtenir une liste réduite des types ICMP pour lesquels il y a un traitement réel. Notez que l'absence de traitements dans le noyau n'empêche pas un programme en espace utilisateur de réagir à l'arrivée de tels messages ; en particulier, certains chevaux de Troie utilisent de tels messages pour prendre connaissance de leurs ordres ou émettre des informations ; il en est de même pour ping, qui nécessite l'ouverture d'un socket_raw afin de lire les réponses (cela explique pourquoi ping est setuid_root). Voici la liste des types ICMP pour lesquels Linux et/ou OpenBSD réagissent en opérant un traitement dans le noyau ou dans les démons liés à IPsec :

Type	Nom	Reference
3	Destination Unreachable	[RFC792]
5	Redirect	[RFC792]
8	Echo	[RFC792]
13	Timestamp	[RFC792]
17	Address Mask Request	[RFC950] (OpenBSD seulement)

Tous les autres types de messages sont ignorés ou interprétés de façon indifférenciée par les applications, dans le cas des messages d'erreurs (par conséquent, il importe peu de faire la différence entre une erreur de type _Source_Quench et une erreur de type _Destination_Unreachable).
Il importe de souligner que notre approche nous permet de retenir les traitements relatifs à ICMP effectués par IPsec sous noyau Linux et sous noyau OpenBSD. Mais des traitements pourraient aussi être menés en espace utilisateur (par exemple par un démon de routage émettant des messages _Router_Advertisement). De tels traitements dépassent le cadre de notre étude.
Soulignons aussi que l'absence de traitements effectués par le noyau en réponse à certains messages n'affecte en rien la granularité des systèmes de filtrages. En particulier, tous les types ICMP décrits dans le fichier header icmp.h sous Linux peuvent être utilisés tels quels pour construire un masque de

filtrage (et, suivant la politique, éventuellement rejeter le message avec un nouveau message, de type `_Destination_Unreachable` et de code `_Communication_Administratively_Prohibited`).

2. Test des Passerelles

La partie précédente nous a permis d'effectuer une étude théorique du comportement des passerelles lors de la réception de messages ICMP. Cette partie va se focaliser sur une étude pratique.
Pendant nos expérimentations, nous avons été amenés à considérer l'utilisation des utilitaires suivants :

* Utilitaires de génération des paquets :

- [hping2]
- [icmpush]
- [nemesis]
- [sendip]
- [sing]

* Utilitaires d'observation du réseau :

- [tcpdump]
- [ethereal]

* Loggers de paquets ICMP :

- pluto/FreeSwan en mode ``DEBUG''
- NetFilter avec la cible ``LOG''
- [icmpinfo]
- [ippl]
- [jail]

Afin de simplifier les manipulations, nous avons classé les types ICMP retenus à l'issue de la première partie (cf. plus haut) en deux catégories, en mentionnant leurs sous-codes :

2.1. Les Messages d'Information

```
Type    Nom                          Reference
----    -------------------------    ---------
 8      Echo                         [RFC792]
          +-Code 0
          `-Code n (n != 0)
13      Timestamp                    [RFC792]
          +-Code 0
          `-Code n (n != 0)
17      Address Mask Request         [RFC950] (OpenBSD seulement)
          `-Code 0
```

Les messages d'information sont relativement faciles à forger de façon automatique. Nous avons donc dressé une liste des adresses réseaux de notre plateforme pouvant provoquer des comportements intéressants avec ICMP (nous n'avons considéré que des adresses IPv4 unicast ou broadcast) :

```
# Adresses Spéciales
NULL_ADDRESS=0.0.0.0
LAN_BROADCAST=255.255.255.255

# Adresses Locales
LOCAL_HOST=127.0.0.1
LOCAL_NET=127.0.0.0
LOCAL_BROADCAST=127.255.255.255

# Réseau 192.168.101.0/24
NET_101_pri=192.168.101.0
BRO_101_pri=192.168.101.255
SGW_101_pri=192.168.101.110
CLI_101_pri=192.168.101.1

# Réseau 192.168.102.0/24
NET_102_pri=192.168.102.0
BRO_102_pri=192.168.102.255
SGW_102_pri=192.168.102.110
CLI_102_pri=192.168.102.1

# Réseau 157.159.101.0/24
NET_101_pub=157.159.101.0
BRO_101_pub=157.159.101.255
SGW_101_pub=157.159.101.110
GW_101=157.159.101.1
PI_101=157.159.101.111

# Réseau 157.159.102.0/24
NET_102_pub=157.159.102.0
BRO_102_pub=157.159.102.255
SGW_102_pub=157.159.102.110
GW_102=157.159.102.1
PI_102=157.159.102.111

# Réseau 157.159.100.0/24
NET_100_pub=157.159.100.0
BRO_100_pub=157.159.100.255
GW_100=157.159.100.3
PI_100=157.159.100.76

# Adresses MAC des passerelles
# (pour avoir la possibilité de
#  ``forcer'' les destinations
#  dans certains tests)
M_GW_100=00:03:E3:53:28:91
M_SGW_101_pub=00:10:5A:45:AD:9A
M_SGW_102_pub=00:60:97:1D:F7:69
```

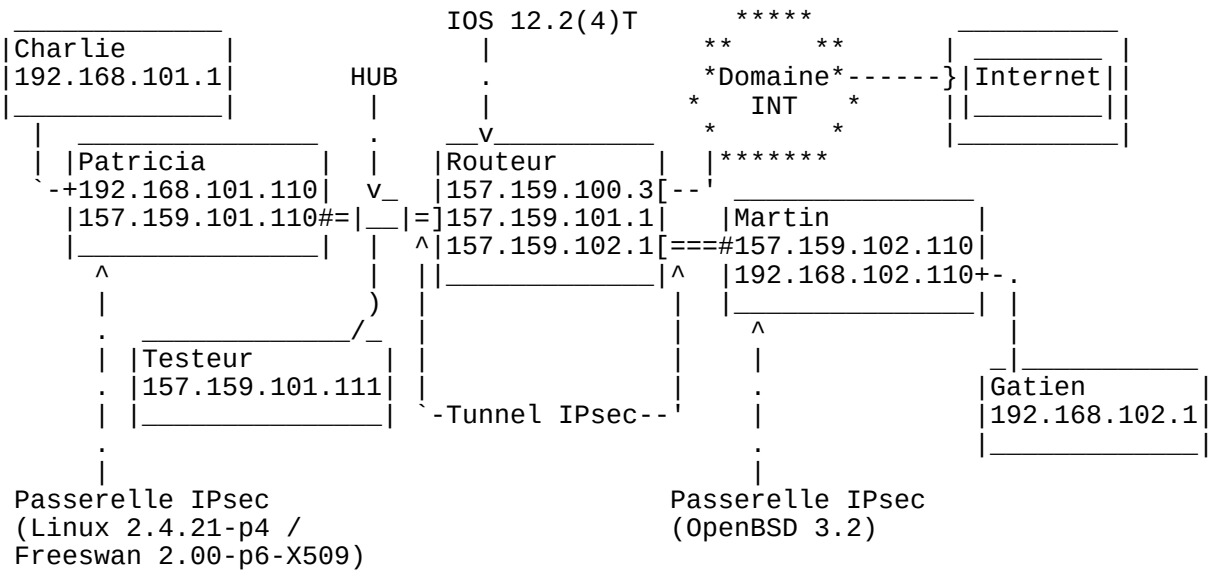
Pour tout couple d'adresses IP (IP1, IP2) tirées dans la liste précédente, et pour toute adresse MAC ``MAC'' de passerelle listée ci-dessus (selon le réseau où sont menés les tests), le script suivant génère un scénario de test complet par passerelle ciblée :

```
#!/bin/sh
for i in `cat ip_addresses.txt`
  do for j in `cat ip_addresses.txt`
    do for k in `cat mac_addresses.txt`
      do
```

```
        echo "nemesis-icmp -i  8 -c 0 -v -S ${IP1} -D ${IP2} -M ${MAC} \
          -d eth0 -\$@" >> $k.script
        echo "sleep 1"  >> $k.script
        echo "nemesis-icmp -i  8 -c 1 -v -S ${IP1} -D ${IP2} -M ${MAC} \
          -d eth0 -\$@" >> $k.script
        echo "sleep 1"  >> $k.script
        echo "nemesis-icmp -i 13 -c 0 -v -S ${IP1} -D ${IP2} -M ${MAC} \
          -d eth0 -\$@" >> $k.script
        echo "sleep 1"  >> $k.script
        echo "nemesis-icmp -i 13 -c 1 -v -S ${IP1} -D ${IP2} -M ${MAC} \
          -d eth0 -\$@" >> $k.script
        echo "sleep 1"  >> $k.script
        echo "nemesis-icmp -i 17 -c 0 -v -S ${IP1} -D ${IP2} -M ${MAC} \
          -d eth0 -\$@" >> $k.script
        echo "sleep 1"  >> $k.script
      done
    done
  done
```

Ces cinq commandes ``nemesis-icmp'' correspondent aux émissions des messages d'information étudiés, avec leurs différents codes.
Les appels à `sleep` entre chaque message généré sont nécessaires pour éviter que les fonctions de limitation du débit des messages dans les noyaux n'affectent les résultats de l'expérience (`sleep` 1 permet d'attendre 1 seconde). Lorsque les messages sont envoyés sans attente, on observe bien une différence entre le nombre de requêtes envoyées, et le nombre de réponses reçues. L'étude préalable des sources nous a donc permis d'éviter ce piège, susceptible de fausser l'interprétation des échanges.
Le script final génère 3 645 paquets pour chacun des trois réseaux testés, soit 10 935 paquets au total (une expérience dure donc plus d'une heure !). Nous avons commencé par le réseau 157.159.101.0/24, dans la configuration suivante :

Tests sur le réseau 157.159.101.0/24



Au cours de cette expérience, les trames Ethernet étaient adressées directement à la passerelle de sécurité.

Résultat des observations :

Observations générales :
Sur notre maquette, aucune réponse n'a jamais été donnée aux messages `_Address_Mask_Request`.
Aucun ``plantage", aucun comportement incohérent ne s'est manifesté sur la maquette.
Nous n'avons observé aucune différence de comportement entre les messages `_Timestamp` et `_Echo` au niveau des PCs. En revanche, le routeur Cisco répond automatiquement aux `_Timestamp` dirigés vers l'adresse de réseau ou l'adresse de broadcast, et les réponses usurpent ces adresses. Bien que particulièrement curieux, ce comportement peut être justifié par une interprétation trop littérale de [Routers-Rqts].
Pendant les manipulations, nous avons aussi fait quelques tests sur d'autres messages ICMP. Un fait remarquable mérite d'être mentionné ici : une seule station a répondu à une requête `_Information_Request`. Il s'agit du plus vieux système de notre réseau (ce message ICMP est obsolète).

Sur Charlie :

- Ne sont arrivés jusqu'à Charlie que les paquets qui lui étaient directement adressés (donc la passerelle n'a pas fait d'erreurs de routage).

- Parmi les adresses sources des paquets acheminés, on ne retrouve aucune trace des adresses suivantes :

```
0.0.0.0
255.255.255.255
127.0.0.0
127.0.0.1
127.255.255.255
157.159.101.0
157.159.101.110
157.159.101.255
192.168.101.0
192.168.101.110
192.168.101.255
```

Par conséquent, on en déduit que la passerelle de sécurité n'assure pas le relais de paquets usurpant les adresses de réseau, de broadcast, ou l'adresse d'une de ses propres interfaces. Cette sécurité est minimaliste, puisque la passerelle pourrait faire de surcroît de l'[Ingress_Filtering].

- Par ailleurs, Charlie a répondu à tous les messages qui lui étaient adressés. Ces messages ont été routés par la passerelle de sécurité (Patricia) en fonction de leurs caractéristiques (et notamment à travers le tunnel ESP).

- Enfin, Charlie a reçu des réponses venant de sa passerelle de sécurité (Patricia). Cette dernière a accepté les paquets usurpés avec comme adresse source celle de Charlie, et a répondu à Charlie. Ici encore, l'utilisation de l'``ingress filtering" aurait été judicieuse. Par ailleurs, Charlie a aussi reçu des réponses de Gatien. Il s'agit des réponses à des paquets émis par le testeur avec pour adresse source Charlie et pour adresse destination Gatien. La passerelle de sécurité a tout naturellement acheminé ces paquets à travers le tunnel vers Gatien.

Sur Gatien :

- Gatien était relativement isolé durant cette expérience et n'a reçu que peu de paquets. Tous ces paquets ont été reçus à travers le tunnel ESP, à l'exception des réponses émises par Martin (voir plus bas).

- En particulier, Gatien a reçu les requêtes usurpant Charlie créées par le testeur et acheminées à travers le tunnel par Patricia. Les requêtes usurpant l'identité de Patricia n'ont pas été acheminées.

- Les véritables réponses de Charlie et de Patricia (sur l'adresse 192.168.101.110) à des requêtes usurpant Gatien ont aussi été acheminées dans le tunnel.

- Les seules autres réponses reçues par Gatien sont celles de Martin, suite aux fausses requêtes construites par le testeur ayant pour source Gatien et pour destination Martin.

Sur Martin :

- Seuls les paquets à destination de Martin ont été observés sur Martin. Il n'y a donc pas eu de problème de routage.

- Parmi les requêtes reçues par Martin, nous n'avons observé aucun `_Address_Mask_Request`. Ces requêtes ne sont donc pas acheminées par le routeur Cisco (Patricia les achemine puisque Gatien a reçu les requêtes de ce type émises avec l'adresse de Charlie à travers le tunnel ESP).

- Le routeur a répondu lorsqu'il était destinataire des paquets (en particulier 157.159.102.110). Il n'a laissé passer aucun paquet ayant pour source un réseau (adresse en .0) ou un broadcast (adresse en .255) sur lequel il a une interface. Il a adopté exactement le même comportement que Patricia à ce niveau.

- Parmi les paquets ayant pour adresse source celle de Charlie (i.e : les seuls paquets du réseau 192.168.101.0/24 que la passerelle a accepté d'acheminer), seuls ceux ayant pour destination Martin sont arrivés en clair à destination. Les autres (à destination d'adresses du réseau 192.168.102.0/24) ont été acheminés à travers le tunnel ESP.

- Pour ce qui est des réponses provoquées par de fausses requêtes, Martin n'a reçu de messages `_Timestamp_Reply` ayant pour source des broadcasts et des réseaux qu'en provenance du routeur (cf. plus haut). Les réponses plus conventionnelles ont été ausi observées.

- Les requêtes reçues par Martin ont évidemment déclenché des réponses de sa part, en particulier à destination de Gatien, du routeur et du testeur. Mais pas à destination de Patricia (rappelons que Patricia n'a pas acheminé les paquets usurpant ses propres adresses source).

Sur Patricia :

- Patricia était l'objet de toute notre attention pendant ce test et les remarques précédentes ont déjà permis de dégager certains de ses traits de comportements.

- En particulier, tous les paquets forgés à destination d'adresses locales (127.*.*.*) ont été traités localement.

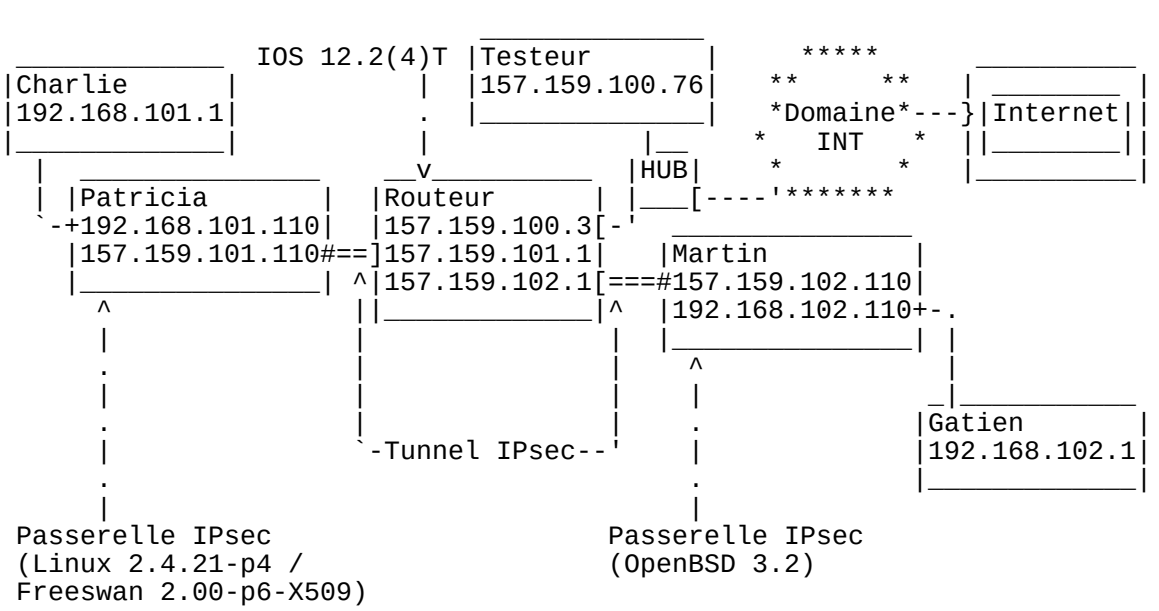
- Nous avons déjà souligné que Patricia a refusé l'acheminement de paquets impliquant des adresses sources de réseau et de broadcast correspondant à des réseaux sur lesquels elle a une interface. Il en est de même pour les paquets usurpant une de ses adresses IP.

- Conformément à ce qui a été dit plus haut, Patricia a acheminé les paquets à destination de 192.168.102.0/24 et depuis 192.168.101.0/24 à travers le tunnel ESP.

- Le trafic généré ici était trop faible pour amener l'association de sécurité à expiration. Cependant, une telle attaque peut être mise en place ; en forçant l'association à expirer, un cracker peut forcer l'exécution de IKE, et provoquer ainsi un déni de service (en épuisant IKE par des fragments UDP). Cette attaque n'étant pas spécifiquement liée à ICMP, il n'en sera pas plus question ici.

Nous avons poursuivi nos expérimentations sur le réseau 157.159.100.0/24, dans la configuration suivante :

Tests sur le réseau 157.159.100.0/24



Au cours de cette expérience, les trames Ethernet étaient adressées directement au routeur.
Résultat des observations :
Observations générales :
Ce scénario présente beaucoup moins d'intérêt vis à vis des passerelles de sécurité. Cependant, il s'avère bien plus réaliste que les deux autres : dans un cas réel, il n'y a peu de chances pour que les paquets ICMP arrivant à la passerelle aient pu conserver des caractéristiques inhérentes à l'aspect local (adresses de réseaux, de broadcast, loopback, etc.).

Sur Charlie :

- Dans ce scénario, Charlie n'a reçu que quatre messages. Il s'agit de deux réponses `_Echo_Reply` et de deux réponses `_Timestamp_Reply`, ayant pour source 157.159.101.110 (Patricia). De tels paquets s'étaient déjà manifestés dans le précédent scénario.

Sur Gatien :

- La situation sur Gatien est théoriquement identique à celle sur Charlie (si ce n'est que les passerelles de ces deux machines tournent sur des OS différents). La pratique n'a pas fait mentir la théorie, et nous avons donc observé quatre messages ayant les mêmes caractéristiques que ceux sus-mentionnés, mais émis depuis 157.159.102.110 (Martin).

Sur Patricia :

- Parmi les réponses ICMP reçues sur Patricia, les seuls paquets particuliers ont été émis par le routeur. Ces paquets sont les _Timestamp_Reply ayant pour source 157.159.101.0, 157.159.101.255, 157.159.102.0, 157.159.102.255, 255.255.255.255, 157.159.100.0 et 157.159.100.255 (nous avons déjà parlé de cette particularité plus haut).

- Par ailleurs, des _Timestamp_Reply et des _Echo_Reply plus classiques ont aussi été reçus, en provenance de 157.159.101.1, de 157.159.102.1 et de 157.159.102.110. La présence de nombreuses réponses aux requêtes ICMP envoyées en broadcast sur le réseau 157.159.100.0/24 (qui est un réseau opérationnel) a rendu l'analyse plus difficile mais n'a provoqué aucun comportement particulier sur la maquette.

- Pour en finir avec les réponses ICMP, Patricia a répondu à toutes les requêtes ayant une source correcte (i.e. différente de celle de Patricia) sur le réseau 157.159.0.0/16, et à Charlie (voir plus haut). Les adresses de la forme 192.168.0.0./16 n'étant pas routables par le routeur Cisco, aucun paquet à destination de Charlie ou de Gatien n'a pu arriver jusqu'à Patricia, et par conséquent, aucun paquet n'est passé par le tunnel ESP.

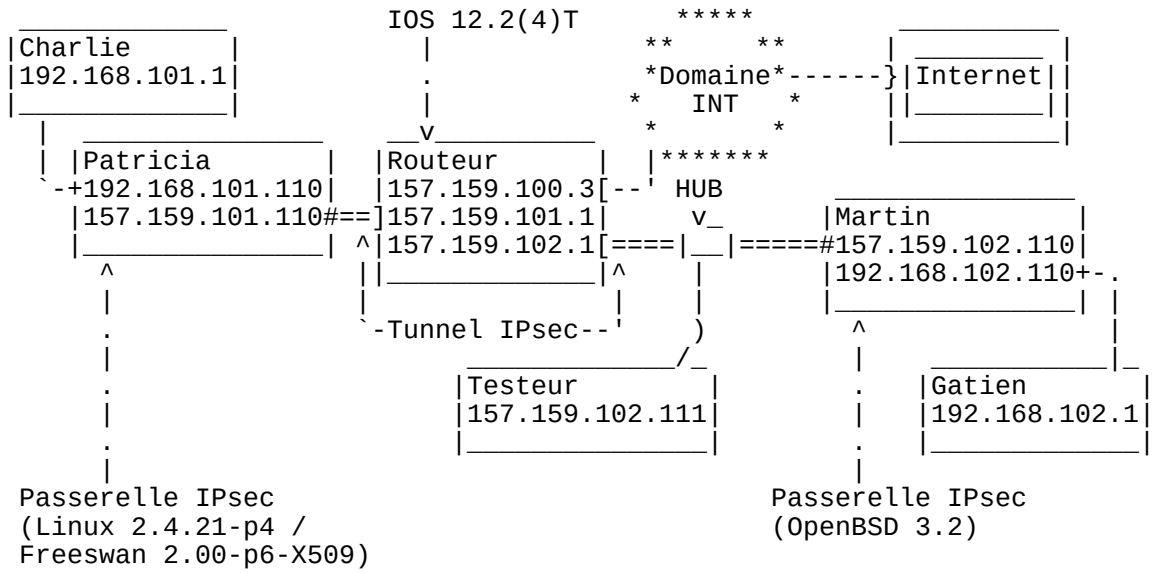
- Enfin, en ce qui concerne les requêtes, absolument rien de particulier ne s'est passé (en dehors du passage des paquets de la forme 192.168.101.1 -> 157.159.101.110 et 192.168.102.1 -> 157.159.102.110 dont il a déjà été question).

Sur Martin :

- Nous avons observé sur Martin exactement les mêmes événements que sur Patricia (aux différences d'adresses près, bien sûr).

Nous avons achevé nos expérimentations sur les messages d'information ICMP avec ce dernier scénario, destiné à tester l'implémentation OpenBSD.

Tests sur le réseau 157.159.102.0/24



Cette fois, les trames Ethernet ont été adressées directement à la machine OpenBSD.
Résultat des observations :
Observations générales :
Dans le contexte présent, le système OpenBSD subit une pression très locale de la part des paquets forgés. En particulier, aucun intermédiaire tel que le routeur ne vient empêcher les paquets d'atteindre leur cible. L'absence de réactions aux requêtes `_Address_Mask_Request` nous amène à conclure que par défaut la station OpenBSD ne répond pas à ces requêtes, bien que sysctl permette d'activer cette fonctionnalité (cf. plus haut).

Sur Charlie :

- Charlie a reçu toutes les requêtes forgées qui lui étaient destinées et ayant une adresse source dans 192.168.102.0/24 :

```
192.168.102.0
192.168.102.1
192.168.102.110
192.168.102.255
```

- Par conséquent, Martin a acheminé à travers le tunnel ces requêtes de la même manière que Patricia dans le premier scénario.

- La différence majeure, que nous avons déjà soulignée plus haut, est que **Martin a acheminé des paquets usurpant une de ses propres adresses (192.168.102.110).**

- Charlie a répondu aux requêtes `_Timestamp` et aux messages `_Echo` vers les adresses sus-mentionnées (donc les réponses ont fait le trajet inverse dans le tunnel).

- Enfin, Charlie a aussi reçu des réponses, consécutives à des requêtes usurpant son adresse comme source. Les réponses ont été faites par l'interface privée de Martin, par Gatien et par Patricia via son interface publique.

Sur Gatien :

- Gatien a acheminé tous les paquets forgés qui lui étaient destinés à l'exception de ceux ayant pour source 0.0.0.0, 127.0.0.0, 127.0.0.1 ou 127.255.255.255. Cette observation confirme un nette différence de fonctionnement entre OpenBSD d'une part et Linux et IOS d'autre part. Cela signifie en particulier qu'une passerelle de type OpenBSD acceptera par défaut de relayer une redirection usurpant sa propre identité si aucune configuration sur PacketFilter n'est effectuée. Par ailleurs, comme dans le cas de FreeSwan, une vérification drastique des politiques de sécurité n'est pas imposée.

- Gatien à donc généré les réponses classiques (i.e : vers des systèmes autres que lui-même et vers des adresses valides), lesquelles ont été acheminées par tunnel si elles étaient à destination du réseau 192.168.101.0/24.

- Enfin, Gatien a aussi reçu les réponses aux messages acheminés par Martin vers Patricia et Charlie et usurpant l'adresse source de Gatien, ainsi que les réponses à des requêtes adressées à Martin sous l'identité de Gatien.

Sur Patricia :

- Les paquets à destination de l'adresse publique de Patricia ont été acheminés par Martin puis par le routeur, sans protection quelconque. Patricia a répondu en clair à toutes les requêtes issues d'adresses publiques différentes de la sienne, et à reçu toutes les réponses issues des mêmes adresses à des requêtes usurpant son identité.

- Les paquets à destination de l'adresse privée de Patricia ou de Charlie ont été encapsulés par Martin dans ESP avant d'être acheminés. Le comportement est tout à fait similaire à celui mentionné ci-dessus, sauf que les acheminements se font dans le tunnel.

Sur Martin :

- Martin a répondu à toute requête ayant pour source une adresse publique, même si cette requête était à destination de son interface privée.

- Par ailleurs, on a pu observer de nombreux messages de redirection émis par Gatien à l'intention de l'adresse usurpée dans les paquets émis. Ce comportement est des plus corrects.

- Martin a reçu des réponses des systèmes suivants :

```
157.159.100.0
157.159.100.3
157.159.100.255
157.159.101.0
157.159.101.1
```

```
157.159.101.110
157.159.102.1
157.159.102.111
192.168.102.1
157.159.101.255 (idem)
```

Les réponses ayant pour source une adresse de réseau ou une adresse de broadcast sont de type `_Timestamp_Reply` et ont été émises par le routeur, conformément à la remarque que nous avons émise dans le premier scénario.

- En dehors des politiques particulièrement laxistes de Martin concernant l'acheminement, nous n'avons rien observé de particulier.

2.2. Les Messages d'Erreurs

```
Type    Nom                                          Reference
----    --------------------------                   ---------
 3      Destination Unreachable                      [RFC792]
           +-Code 0  _Net_Unreachable
           +-Code 1  _Host_Unreachable
           +-Code 2  _Protocol_Unreachable
           +-Code 3  _Port_Unreachable
           +-Code 4  _Fragmentation_Needed_and_Don't_Fragment_was_Set
           +-Code 5  _Source_Route_Failed
           +-Code 6  _Destination_Network_Unknown
           +-Code 7  _Destination_Host_Unknown
           +-Code 8  _Source_Host_Isolated
           +-Code 9  _Communication_with_Destination_Network_is_
                      Administratively_Prohibited
           +-Code 10  _Communication_with_Destination_Host_is_
                       Administratively_Prohibited
           +-Code 11  _Destination_Network_Unreachable_for_Type_of_Service
           +-Code 12  _Destination_Host_Unreachable_for_Type_of_Service
           +-Code 13  _Communication_Administratively_Prohibited [RFC1812]
           +-Code 14  _Host_Precedence_Violation                 [RFC1812]
           `-Code 15  _Precedence_cutoff_in_effect               [RFC1812]

 5      Redirect                                     [RFC792]
          +-Code 0  _Redirect_Datagram_for_the_Network_(or_subnet)
          +-Code 1  _Redirect_Datagram_for_the_Host
          +-Code 2  _Redirect_Datagram_for_the_Type_of_Service_and_Network
          `-Code 3  _Redirect_Datagram_for_the_Type_of_Service_and_Host
```

Faute de temps, nous n'avons pu mener cette partie de l'étude.

3. Les Attaques via ICMP

3.1. Dénis / Dégradations de Services

Smurf

Le smurf est une attaque visant à saturer un hôte de paquets ICMP. Pour ce faire, le cracker émet des requêtes ICMP, par exemple du type _Echo, en usurpant l'adresse source de la victime, et en les envoyant vers une adresse de broadcast. Si le routeur en charge de l'adresse de broadcast assure la

translation vers une adresse de broadcast de niveau lien (par exemple un broadcast Ethernet), alors de nombreux systèmes peuvent répondre à ces requêtes, et noyer la victime. Cette attaque est plus efficace avec des paquets de taille importante (64 Ko).

Ce type d'attaque fait désormais preuve d'une efficacité limitée, pour plusieurs raisons :

- La plupart des systèmes d'exploitation actuels, dans leur configuration par défaut, ne répondent pas à des requêtes émises en broadcast. Linux est cependant une exception, puisque par défaut il accepte de répondre au broadcast (mais avec les contraintes ci-dessous).

- Les noyaux actuels limitent les débits d'émission des messages ICMP (cf. l'analyse des sources, plus haut). Le facteur d'amplification de l'attaque est donc réduit.

Nous avons voulu vérifier ces points dans les faits. Concernant les réponses aux messages adressés au broadcast, sur un réseau opérationnel de classe C, nous avons enregistré des réponses de 74 systèmes différents. Une partie de ces systèmes sont effectivement des systèmes Linux, mais les autres sont plus diversifiés. Cela prouve qu'un réseau opérationnel peut avoir un existant relativement lourd.
Par ailleurs, concernant l'effet d'amplification du smurf, la pratique n'a pas fait mentir la théorie. Il s'est avéré en fait plus efficace de noyer une victime sous un burst de requêtes que d'espérer de nombreuses réponses à l'issue de l'émission sur le broadcast ! Dans tous les cas, la victime n'a pas souffert des attaques (aucune dégradation visible des performances, tant au niveau des besoins de fragmentation, qu'au niveau du débit ou des délais d'expiration des connexions).
Par ailleurs, nous avons constaté que le routeur local adopte un comportement très particulier, qui nous était inconnu, lorsque l'on tente d'émettre des requêtes vers un broadcast sur un autre réseau... En effet, dans ce cas, c'est lui-même qui répond, avec sa propre adresse !
Nota : une variante de smurf, appelée ``fraggle'', se base sur UDP à la place d'ICMP, et utilise par exemple le service ``echo'' (port UDP N°7).

Dégradation de Service

L'objectif initial du smurf, présenté précédemment, est d'aboutir à un déni de service. Actuellement, dans le meilleur des cas (pour le cracker), le smurf aboutira à une détérioration du service.
Le cracker peut alors rechercher un moyen plus discret d'arriver aux mêmes effets...
Pour ce faire, il peut se servir du fait que la plupart des systèmes émettent des paquets avec le bit `Don't_Fragment` actif et mettent en œuvre le mécanisme de [PMTU]. Le cracker envoie alors des messages `_Destination_Unreachable` / `_Fragmentation_Needed_and_Don't_Fragment_was_Set`, en annonçant un MTU plus faible que nécessaire. Ce faisant, il peut réduire de façon virtuelle la bande passante disponible sur un lien, voire même provoquer un déni de service sur les paquets tunnelés.
Cette technique présente deux avantages :

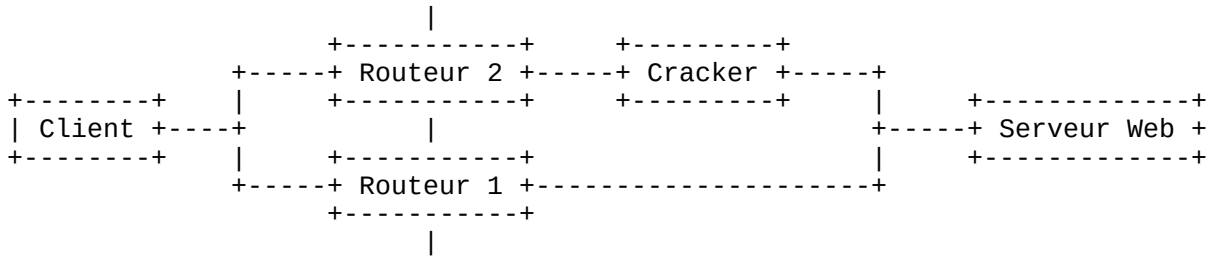
- Il n'est pas nécessaire d'attaquer directement les extrémités (on peut s'en prendre aux passerelles intermédiaires), ce qui rend difficile un diagnostic.

- La dégradation du service peut s'opérer d'une manière relativement continue. Le cracker peut ainsi donner l'impression à l'utilisateur final que les performances du réseau sont en cause, et non qu'il est victime d'une attaque. L'utilisateur final, lassé, rompra la connexion, et le résultat sera donc un déni de service sur l'utilisateur et non sur le système.

D'autres attaques visant à dégrader le service pourraient se baser par exemple sur les messages _Source_Quench. Ces messages n'étant pas pris en compte par la plupart des systèmes, nous n'avons pas poussé l'investigation plus loin.

3.2. Détournement de connexions

Ces attaques se basent sur l'utilisation des messages _Redirect. Soulignons d'avance qu'une redirection vers un routeur inexistant peut provoquer un déni de service (le cracker doit cependant s'assurer périodiquement que la redirection est encore effective).
Nous n'avons pas eu suffisamment de temps pour procéder à tous les tests nécessaires sur les redirections dans le chapitre précédent, mais nous avons tout de même procédé à une petite expérience entre deux hôtes. La redirection a parfaitement été prise en compte, nous permettant de détourner un trafic http. En utilisant ces messages, un cracker devrait être en mesure de mener la plupart des attaques contre IPsec précédemment décrites dans notre rapport portant sur ARP (ces attaques se basent notamment sur le laxisme des politiques de sécurité). Selon le contexte, le cracker pourra sélectionner la méthode la plus discrète (cela dépend en particulier des capacités d'observation des commutateurs locaux).
En théorie, le cracker n'est pas contraint d'effectuer une attaque par redirection localement. Cependant, concrètement, de tels messages ont des probabilités élevées d'être filtrés. Par ailleurs, l'intérêt de l'attaque peut dépendre beaucoup de la topologie. Par exemple, considérons le schéma suivant :



Dans ce scénario, le client utilise le Routeur 1 comme route par défaut. Grâce à une redirection, le cracker amène le client à croire qu'il doit passer par le Routeur 2 pour atteindre le serveur Web, ce qui lui permet d'intercepter le trafic.
On serait tenté de supposer, par application récurrente de la manipulation précédente, que le cracker peut rediriger jusqu'à lui le trafic d'une victime à destination d'un serveur en créant une route spécifique, par empoisonnement successif de différents routeurs intermédiaires, en leur annonçant une route appropriée vers le serveur. Cependant, ce qui fonctionne pour les hôtes ne devrait pas, en conformité avec [Routers-Rqts], fonctionner pour les routeurs :

Extrait du § 5.2.7.2 de [Routers-Rqts] :

```
A router using a routing protocol (other than static routes) MUST NOT
consider paths learned from ICMP Redirects when forwarding a packet.
If a router is not using a routing protocol, a router MAY have a
configuration that, if set, allows the router to consider routes
learned through ICMP Redirects when forwarding packets.
```

Cela diminue grandement la portée des attaques via _Redirect... Par ailleurs, de nombreuses vérifications peuvent être effectuées sur les messages de redirection acheminés : les adresses source et

destination doivent appartenir au même réseau et être accessibles sur la même interface, l'adresse source doit être un routeur de la zone, etc. La mise en œuvre de l'[Ingress_Filtering] doit être suffisante pour contrer la plupart des attaques distantes. Le refus systématique d'acheminer et de prendre en compte les messages `_Redirect` sur les routeurs semble cependant la meilleure solution.
Remarque : des problèmes se sont manifestés lors de la génération des messages `_Redirect` sous Linux. En effet, les derniers ``hook'' de la chaîne de sortie des paquets opèrent les post-traitements de translation d'adresse. Or, dans le cadre de la translation d'adresse, l'acheminement des redirections est proscrit. Par conséquent, les tests mettant en oeuvre des redirections nécessitent d'abandonner la translation d'adresse. Nous avons pris contact avec d'autres personnes intéressées pour proposer un patch pour le noyau afin de pouvoir régler ce problème.

3.3. Scan ICMP

Les messages ICMP générés par les systèmes constituent une source non négligeable d'informations pour les crackers.
Les messages d'information apportent des renseignements sur le système interrogé (horloge système, masque de sous-réseau, etc.).
Les messages d'erreur permettent d'obtenir plusieurs informations qui sont à la base de diverses techniques de scan. Plus précisément, les messages `Destination_Unreachable` sont au premier plan, puisqu'ils permettent de déterminer si des ports sont ouverts, si un protocole est installé sur un système, si des hôtes ou des réseaux sont accessibles, etc. Par ailleurs, les ``traceroutes'' se basent aussi sur ICMP pour donner une idée des routes empruntées. Des envois de fragments permettent aussi de repérer des pare-feux-proxy transparents, puisque ces derniers reconstituent en général les paquets avant de les analyser, et émettent, maladroitement, des erreurs `_Time_Exceeded` / `_Fragment_Reassembly_Time_Exceeded` lorsqu'ils ne peuvent reconstituer le paquet.
Des outils tels que [firewalk] mettent en oeuvre certaines de ces techniques pour tenter de déterminer les règles d'un pare-feux.
Ce sujet, ainsi que le suivant, a été abondamment (220 pages) traité par Ofir Arkin dans [ICMP-Scan]. Cette publication faisant référence, nous engageons le lecteur à s'y reporter pour plus de renseignements.

3.4. Prise d'Empreintes de Systèmes d'Exploitation

Les RFCs ne spécifient pas de façon exhaustive le comportement d'un protocole ou les valeurs d'un champ dans un message. Les développeurs sont confrontés à de nombreux choix dont certains sont guidés par des considération pratiques relatives à l'implémentation, ou par la subjectivité.
Les implémentations des piles IP ont donc des personnalités, de même que les valeurs choisies dans les champs complémentaires des messages ICMP.
Le document [ICMP-Scan] mentionne différentes caractéristiques spécifiques à divers systèmes d'exploitation dans les messages ICMP. Le programme [xprobe], que nous utilisons régulièrement, implémente ces méthodes et témoigne de leur efficacité pour déterminer le système d'exploitation d'un hôte distant.

4. Précautions à prendre vis à vis d'ICMP

4.1. L'Observation

Les différents éléments constitutifs des services réseaux d'un système proposent souvent des modes de journalisation. De telles fonctionnalités sont ainsi directement intégrées dans **NetFilter** et pluto. **NetFilter** dispose en outre de la capacité d'intercepter des paquets incorrects.
Par ailleurs, nous avons mentionné précédemment, lors des manipulations, l'existence de nombreux logiciels spécifiques assurant l'observation des messages ICMP.
Enfin, l'utilisation d'IDS plus génériques et possiblement externes au système étudié est envisageable.
Malheureusement, dans tous les cas, ces systèmes nécessitent des processus d'interprétation et des mesures d'action en cas d'alarme. Ces considérations sont actuellement laissées trop souvent de côté.

4.2. Durcissement du noyau

Un durcissement du noyau est tout à fait possible afin de limiter les risques. Le cas extrême consisterait à ``sabrer'' certaines fonctions appelées lors de la réception d'un message ICMP (il n'est pas réaliste de se passer de toutes ces fonctions), et de rendre inefficace la fonction d'émission de messages ICMP pour presque tous les types de messages (cf. partie suivante pour le traitement particulier du PMTU).
Plusieurs correctifs sont aussi disponibles sur différents sites pour réduire les risques liés à ICMP. Nous avons aussi présenté plus haut les possibilités de configuration préalables à la compilation.
Enfin, affiner la configuration à l'exécution via sysctl peut être préférable. Par exemple :

Extraits de /etc/sysctl.conf :

```
net/ipv4/icmp_echo_ignore_all=1
net/ipv4/icmp_echo_ignore_broadcasts=1
net/ipv4/icmp_ratemask=524288
net/ipv4/conf/eth0/accept_redirects=0
net/ipv4/conf/eth0/secure_redirects=0
net/ipv4/conf/eth0/send_redirects=0
```

Résultats des tentatives de scan sur un système configuré tel que ci-dessus :

```
cerbere:~# nmap ivan.int-evry.fr
Starting nmap 3.27 ( www.insecure.org/nmap/ ) at 2003-06-05 21:26 CEST
Note: Host seems down. If it is really up, but blocking our ping probes, try -P0
Nmap run completed -- 1 IP address (0 hosts up) scanned in 12.307 seconds
cerbere:~# nmap -P0 ivan.int-evry.fr
Starting nmap 3.27 ( www.insecure.org/nmap/ ) at 2003-06-05 21:37 CEST
All 1623 scanned ports on ivan.int-evry.fr (157.159.100.48) are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned in 87.219 seconds
cerbere:~# xprobe2 ivan.int-evry.fr
XProbe2   v.0.1   Copyright   (c)   2002-2003   fygrave@tigerteam.net,   ofir@sys-
security.com, meder@areopag.net
[+] Target is ivan.int-evry.fr
[+] Loading modules.
[+] Following modules are loaded:
        [x]ICMP echo (ping)
        [x]TTL distance
```

```
        [x]ICMP echo
        [x]ICMP Timestamp
        [x]ICMP Address
        [x]ICMP Info Request
        [x]ICMP port unreach
[+] 7 modules registered
[+] Initializing scan engine
[+] Running scan engine
[+] Host: 157.159.100.48 is down (Guess probability: 0%)
[+] Cleaning up scan engine
[+] Modules deinitialized
[+] Execution completed.
cerbere:~#
```

4.3. Filtrage

Dans de nombreux cas, des méthodes de sécurité dites ``topologiques'' constituent une aide appréciable en ce qui concerne la validation des messages ICMP autorisés à circuler sur le réseau local. Ces types de traitement peuvent être opérés par des routeurs filtrants ou des pare-feux. Il existe de plus des fonctionnalités de normalisation des paquets (harmonisation des TTL, etc.) et de suivi de connexion (vérification de la légitimité des messages) sur certains de ces systèmes.
De façon générale, il est important de réaliser que ICMP est un protocole dont le rôle est principalement de fournir des informations complémentaires relatives à une situation donnée. En particulier, une implémentation correcte d'une pile de protocoles Internet doit pouvoir se passer d'ICMP. D'ailleurs, aucune garantie n'est jamais donnée sur la délivrabilité d'un tel message, et IP et ICMP ont été conçus dans la pleine conscience de cette considération. Cela ne signifie cependant pas qu'il faut filtrer ICMP d'une façon outrancière. D'un point de vue général, un client classique (web, mail, IPsec) doit pouvoir se passer d'émettre ou de recevoir des messages autres que `_Destination_Unreachable`. Ces derniers sont en effet très importants pour un fonctionnement correct de [PMTU].

4.4. Politiques de Sécurité

Dans le cadre de l'utilisation d'IPsec, nous avons pu constater que les implémentations testées n'appliquent qu'à la légère les politiques de sécurité que le standard requiert pourtant.
En l'absence d'une application forte de ces politiques, les systèmes de filtrage sus-mentionnés sont **indispensables**.

5. Conclusion

Il ressort de notre étude que l'importance généralement accordée à ICMP est très surfaite. D'une part les messages ICMP sont loin d'être tous traités par les systèmes, d'autre part, l'information qui en découle n'est pas indispensable, et dans certains cas, peut même s'avérer inexploitable.
Nous pouvons donc préconiser un filtrage ``dur'' de ICMP au niveau de l'adhérence d'un réseau. Un tel filtrage nécessite de connaître précisément quels sont les types ICMP importants. Dans la plupart des cas, seuls les messages nécessaires au bon fonctionnement du [PMTU] sont indispensables.
En pratique, l'utilisation conjointe d'IPsec et d'ICMP ne pose aucun problèmes. En effet, les implémentations ont volontairement omis de prendre des risques. D'un point de vue théorique, quelques points méritent cependant un approfondissement, en particulier le redimensionnement des interfaces d'un tunnel, l'interprétation des messages _Destination_Unreachable comme indices possibles de déconnexion des hôtes distants, l'utilisation (active) de messages d'information ICMP pour déterminer l'état de l'hôte distant, et l'affectation d'une mesure de confiance à des messages d'erreurs reçus par des routeurs intermédiaires.

6. Références

[ARP] **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)

[ethereal] **Ethereal**
Status : Programme
Auteur(s) : G. Combs
(Source : http://www.ethereal.com)

[firewalk] **FireWalk**
Status : Programme
Auteur(s) : M. Schiffman
(Source : http://http://www.packetfactory.net/firewalk/)

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

[Hosts-Rqts] **Requirements for Internet Hosts -- Communication Layers**
Date : Octobre 1989
Status : RFC 1122 (Standard)
Auteur(s) : R. Braden
(Source : http://www.ietf.org/rfc/rfc1122.txt)

[hping2] **HPing II**
Status : Programme
Auteur(s) : S. Sanfilippo
(Source : http://www.hping.org)

[IANA] **Internet Assigned Numbers Authority**
Status : Organisation
(Source : http://www.iana.org)

[ICMP] **Internet Control Message Protocol**
Date : 1er Septembre 1981
Status : RFC 792 (Standard)
Auteur(s) : J. Postel
(Source : http://www.ietf.org/rfc/rfc792.txt)

[ICMP-Scan] **ICMP Usage in Scanning**
Date : Juin 2001
Status : Rapport
Auteur(s) : O. Arkin
(Source : http://www.sys-

security.com/archive/papers/ICMP_Scanning_v3.0.pdf)

[icmpinfo] **ICMPInfo**
Status : Programme
Auteur(s) : L. Demailly
(Source : http://www.demailly.com/~dl/softs.html)

[icmpush] **ICMPush**
Status : Programme
Auteur(s) : Slayer, !H
(Source : http://hispahack.ccc.de)

[IKE] **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)

[Ingress_Filtering] **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)

[ippl] **IP Protocols Logger**
Status : Programme
Auteur(s) : H. Haas, E. Bernard
(Source : http://pltplp.net/ippl/)

[jail] **Just Another ICMP Logger**
Status : Programme
Auteur(s) : A. Menon-Sen
(Source : http://www.wiw.org/~ams/jail/)

[Linux] **Linux**
Status : Système d'Exploitation Libre
Auteur(s) : L. Torvalds & many contributors
(Source : http://www.kernel.org)

[MIB-TCP/IP] **Management Information Base for Network Management of TCP/IP-based internets**
Date : Mai 1990
Status : RFC 1156 (Informational)
Auteur(s) : K. McCloghrie, M. Rose
(Source : http://www.ietf.org/rfc/rfc1156.txt)

[nemesis] **Nemesis**
Status : Programme
Auteur(s) : M. Grimes, J. Nathan
(Source : http://www.packetfactory.net/Projects/nemesis/)

[OpenBSD]

OpenBSD
Status : Système d'Exploitation Ouvert
(Source : http://www.openbsd.org)

[PMTU]

Path MTU Discovery
Date : Novembre 1990
Status : RFC 1191
Auteur(s) : J. Mogul, S. Deering
(Source : http://www.ietf.org/rfc/rfc1191.txt)

[Routers-Rqts]

Requirements for IP Version 4 Routers
Date : Juin 1995
Status : RFC 1812 (Proposed Standard)
Auteur(s) : F. Baker
(Source : http://www.ietf.org/rfc/rfc1812.txt)

[sendip]

SendIP
Status : Programme
Auteur(s) : M. Ricketts
(Source : http://www.earth.li/projectpurple/progs/sendip.html)

[sing]

Send ICMP Nasty Garbage
Status : Programme
Auteur(s) : A. A. Omella
(Source : http://sourceforge.net/projects/sing/)

[tcpdump]

TCPDump
Status : Programme
(Source : http://www.tcpdump.org)

[xprobe]

Xprobe
Status : Programme
Auteur(s) : F. Yarochkin, O. Arkin
(Source : http://www.sys-security.com/html/projects/X.html)