RFC 5201 : Host Identity Protocol

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 21 avril 2008

Date de publication du RFC : Avril 2008

https://www.bortzmeyer.org/5201.html

HIP, décrit dans ce RFC, est un protocole très ambitieux, puisqu'il vise à compléter IP en fournissant une séparation de l'identificateur et du localisateur <https://www.bortzmeyer.org/separation-identificateur.html>, permettant d'améliorer la sécurité (notamment la résistance aux DoS) et de mieux gérer la mobilité et le "*multi-homing*". La version 1, décrite dans ce RFC, a depuis été remplacée par la v2, dans le RFC 7401 [1].

L'architecture générale de HIP est décrite dans le RFC 9063. Notre RFC normalise, lui, le protocole concret. HIP repose d'abord sur la séparation entre un nouvel **identificateur**, le HI ("*Host Identifier*") et un **localisateur**, plus concret, qui sera simplement l'adresse IP, réduite à un rôle moins visible, sans exigence de stabilité. Par exemple, HIP permettra le changement de localisateur (d'adresse IP) en cours de connexion, sans rompre celle-ci, ce qui sera précieux pour la mobilité.

HIP est donc déployable uniquement en modifiant les machines terminales du réseau (si les coupe-feux le laissent passer), sans toucher aux routeurs. Il en existe des mises en œuvres pour FreeBSD <http://hip4inter.net/> et Linux <http://www.openhip.org/>. Le projet OpenHIP <http://www.openhip.org/> adapte également des logiciels comme Wireshark pour qu'ils aient un support HIP. InfraHIP <http://infrahip.hiit.fi/> travaille également à l'infrastructure HIP. L'article "*HIP Implementation for FreeBSD*" <http://www.hip4inter.net/documentation/hip4bsd.pdf> donne des détails intéressants sur la mise en œuvre de HIP, notamment sur les questions liées à l'API.

Si on ne veut pas lire le RFC 9063, on peut néanmoins avoir une bonne introduction à HIP en lisant les sections 1 et 2 de ce RFC. Elles expliquent le vocabulaire (par exemple, HIP, étant un protocole situé

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc7401.txt

en haut de la couche 3 n'utilise pas le terme de connexion mais celui d'**association**), le nouvel espace de nommage et les principes du protocole.

L'espace de nommage fait l'objet de la section 3. On distingue les HI ("*Host Identifier*"), qui sont des **clés publiques** d'un couple clé privée / clé publique et qui sont de taille variable, et les HIT ("*Host Identifier Tag*", décrits dans la section 3.1), qui sont un résumé cryptographique des HI. Ils sont de taille fixe (donc plus faciles à traiter), 128 bits, la taille d'une adresse IPv6. Un préfixe ORCHID (RFC 4843), le `2001:10::/28`, sert à éviter toute collision avec les « vraies » adresses IPv6. Avec OpenHIP, la clé peut être générée par le programme `hitgen` qui fabrique un fichier XML ressemblant à ceci :

```
<?xml version="1.0" encoding="UTF-8"?>
<my_host_identities>
  <host_identity alg="RSA" alg_id="5" length="128" anon="no" incoming="yes" r1count="10">
    <name>horcrux-1024</name>
    <N>C6EBA2894C33A1312B38853A8ECC0D7967496237A65529807EDF23C4DA753EE88F8FBF71BE38B6910181D5B75DB075B996232
    <E>010001</E>
    <D>383A51165838DBDE872611DACC94775692D09677BE87A214954843D7181D3E2C04B0905FF9721481069909AD2C497DED78B7F
    <P>FF03F93454C9C2D8EC47FE8C9DBF0D82F05E13905F304A5ACA42C45E579F917B4C8CEFEF6B06AAB9BCB7A911D5514B7AEE685
    <Q>C7B0394EB5506B2B75E19E5654262E843659BB76A465C2A7AC47A430749944378E3161FF805B4C6CB037B5CB111F0EF49FF03
    <dmp1>7426C128DEBD8EEBF2A2D004080D6F0006AF32C5FD352788B6BB3669AA0B59DE08FDE082F202755C67E25735722DB6ED65
    <dmq1>1B97DE5361FA9AD4869586ABA7351F78658A40BD443A4B8B9FE2C66D6BAF421DEB2827C2869A17156DC444FAAA83002E0D
    <iqmp>7499A27F59CA1746F1A6E5DE832592F8ACF80B814DD511C490614C44DC92B5CD1650AC944ED5751F28846487C221E8C17F
    <HIT>2001:1f:cd4:7125:2427:f77c:d1b6:e15f</HIT>
    <LSI>1.182.225.95</LSI>
  </host_identity>
</my_host_identities>
```

Notez bien que le fait d'utiliser XML est un choix de OpenHIP, qui n'est utilisé qu'en local. Il n'est pas imposé par la norme qui, sur le câble, n'utilise que du binaire. Les éléments comme `P`, `Q` ou `iqmp` sont les éléments d'une clé RSA (HIP peut utiliser d'autres formats que RSA). Le HIT est représenté en utilisant la syntaxe des adresses IPv6, puisqu'il a la même taille et a été conçu pour être stocké comme une adresse par les applications.

La section 3.2 explique comment générer un HIT à partir du HI. Étant un résumé cryptographique (fait avec SHA-1), il est sûr, on ne peut pas fabriquer facilement un HI qui aurait le même HIT.

Pour signer les paquets, les deux machines utiliseront au début un échange de Diffie-Hellman.

La section 4 donne une vision générale du protocole, qui sera ensuite détaillée dans les sections ultérieures. HIP a reçu le numéro de protocole <https://www.iana.org/assignments/protocol-numbers> 139.

La section 4.1 décrit comment former une **association** entre deux machines HIP. Celle qui demande l'association est nommée l'**initiateur**, celle qui l'accepte le **répondeur**. Le protocole d'association nécessite quatre paquets. En effet, avec seulement trois paquets, comme le fait TCP (RFC 793) lors de l'établissement d'une connexion ("*Three-way handshake*"), on ne peut pas à la fois se protéger contre les DoS et permettre des options par connexion. Se protéger contre les DoS nécessite de ne **pas** garder d'état tant que le pair n'est pas authentifié, même faiblement. Les techniques qui permettent à TCP de ne pas garder d'état sur le « répondeur », telles que les "*SYN cookies*" du RFC 4987 sont incompatibles avec les options TCP.

Voici pourquoi les protocoles plus récents comme SCTP (RFC 3286) ou comme HIP nécessitent quatre paquets.

Dans HIP, ils sont nommés **I1**, **R1**, **I2** et **R2**. Les paquets I sont envoyés par l'initiateur et les paquets R par le répondeur.

L'établissement d'une association se passe donc comme ceci :
— L'initiateur envoie I1 au répondeur. Le paquet contient l'identificateur (le HIT) de l'initiateur et (optionnellement) celui du répondeur. Aucun état n'est créé chez le répondeur.
— Le répondeur envoie R1. Ce paquet contient un puzzle cryptographique que l'initiateur devra résoudre. Il contient également les paramètres de la session Diffie-Hellman. Et ce paquet est signé.
— L'initiateur envoie I2, qui contient la solution du puzzle. les paramètres Diffie-Hellman de l'initiateur. Ce paquet est signé.
— Le répondeur envoie R2 pour accepter l'association. Ce paquet est signé.

Le puzzle, détaillé en section 4.1.1, est un petit problème de calcul que l'initiateur doit résoudre pour montrer qu'il est près à « payer », à faire un effort pour que l'association soit acceptée. La difficulté du puzzle peut être réglée par le répondeur. En pratique, il est très difficile d'imaginer un puzzle qui soit dissuasif contre des attaquants disposant d'un "*botnet*" entier, tout en étant soluble par des appareils simples, genre téléphone portable, ne possédant guère de puissance de calcul. (La section 4.1.1 détaille ce problème et les solutions possibles.)

La section 4.1.3 détaille l'utilisation du Diffie-Hellman.

Une des grandes forces de HIP est la possibilité de **mettre à jour** une association existante (section 4.2). À tout moment, pendant la session, un paquet de type UPDATE peut être envoyé pour changer certains paramètres de la session, comme les localisateurs (les adresses IP) utilisées. La signature des paquets permet de s'assurer que le paquet UPDATE est authentique.

La section 5 est longue car elle détaille le format, bit par bit, de tous les paquets échangés. Il y a huit types de paquets (section 5.3) comme I1, R1, I2, R2 ou comme UPDATE, présenté plus haut.

Enfin, la section 6 détaille le traitement des paquets, ce qu'il faut faire en les recevant, les erreurs et la façon de les gérer (règle simple : ne pas renvoyer de paquets HIP en cas d'anomalie, seulement des ICMP, et avec un débit limité, car on ne peut pas être sûr du pair s'il y a une erreur), etc.