

# HIP Implementation for FreeBSD

hip@hip4inter.net

November 2, 2005

# Contents

<b>1</b>	<b>Host Identity Protocol</b>	<b>1</b>
1.1	Location and Identity of a Host . . . . .	1
1.2	Base exchange and SA establishment . . . . .	2
1.3	Mobility and Multihoming . . . . .	2
<b>2</b>	<b>HIP4BSD Implementation</b>	<b>3</b>
2.1	FreeBSD . . . . .	3
2.2	Mac OS-X . . . . .	4
2.3	Linux . . . . .	4
<b>3</b>	<b>Installing and Testing HIP</b>	<b>5</b>
3.1	Installing HIP . . . . .	5
3.2	Testing the Installation . . . . .	5
<b>4</b>	<b>Future work</b>	<b>7</b>

# 1 Host Identity Protocol

Host Identity Protocol (HIP) is currently being specified in Internet Engineering Task Force (IETF) HIP Working Group:

<http://www.ietf.org/html.charters/hip-charter.html>

## 1.1 Location and Identity of a Host

The current protocols in the Internet use the IP address both as the identifier and locator of a host. This causes some drawbacks e.g. in mobility management. One way to solve this is to separate these two pieces of information from each other. In HIP, the separation is done by introducing a new name space, the Host Identity.

Host Identities are cryptographic in nature. In fact, they are public keys of a public-private key pairs. Thus the owner of the private key can prove that it owns the public key, i.e. the identity. The binding between the identity and the location providing IP address is dynamic and can be changed when the host changes its location. Still, the identity remains the same.

Figure 1 shows the new layer in the stack. In this layer, mappings between Host Identities and IP addresses are handled. The layer hides the IP addresses from the upper layers, where only the Host Identity, or actually its shorter representative Host Identity Tag (HIT) is shown. The HIT is a 128 bits long value, containing a hash over the Host Identity. It can be used directly over the IPv6 API and applications cannot see the difference between a HIT and an IPv6 address.

When IPv4 API is used, HIT cannot be used due to its size. Therefore, a shorter format of identifier, Local Scope Identifier (LSI) is defined, that is 32 bits long. The LSI is handled only locally and it is not visible externally. The Socket API makes the local mapping between the LSI and the HIT for further processing.

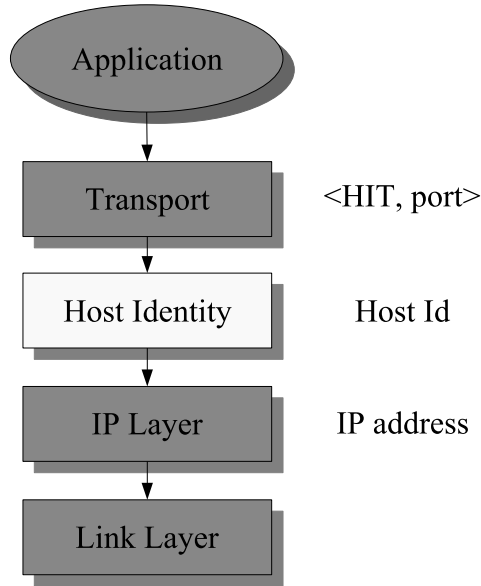


Figure 1: Host stack

## 1.2 Base exchange and SA establishment

The HIP base specification [4] defines the HIP base exchange, during which the end-hosts are authenticated and security parameters are exchanged for creating a shared keying material.

The “Using ESP transport format with HIP” [2] specification defines how the ESP encryption is used for user data protection after HIP negotiation.

## 1.3 Mobility and Multihoming

A host is mobile when it changes its topological location in the Internet. Changing the location means also that the IP address of the host changes.

A host is multihomed when it has more than one simultaneous connections to the Internet. I.e. it has multiple routes to use for outgoing traffic and it can receive traffic using different interfaces.

Because the mapping between the Host Identity and the IP address is dynamic, the locator can be changed and a new binding between the HI and the IP address is generated. The Mobility and Multihoming specification [ref], defines this procedure in more details both for mobility and multihoming cases.

## 2 HIP4BSD Implementation

This implementation is based on the following drafts: Host Identity Protocol Architecture [3], Host Identity Protocol [4], End-Host Mobility and Multi-homing with the Host Identity Protocol [1], Using ESP transport format with HIP [2], and A Bound End-to-End Tunnel (BEET) mode for ESP [6].

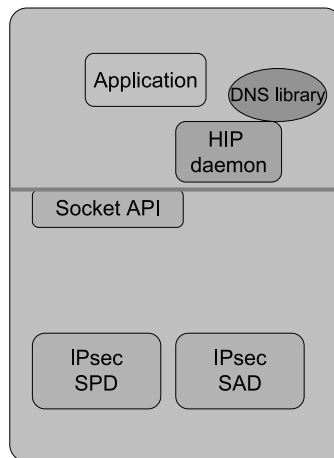


Figure 2: Affected components at a host

### 2.1 FreeBSD

We selected FreeBSD as the main development environment. The FreeBSD implementation contains modifications that are needed to resolve the HITs, to dump the BEET security associations (setkey), and naturally to run the base exchange (hip daemon). Currently supported FreeBSD versions are: FreeBSD-5.4 and FreeBSD-6.0. (See the latest information from our web page <http://hip4inter.net>)

Figure 2 shows the components that have been modified in the implementation. Modifications include:

- HIP daemon

The HIP daemon handles all the main activities of HIP:

- HIP State machine handling as described in base [4] draft
- HIP packet generation and handling

- HIT to address mappings
- HIP packet communication between the user level and the kernel
- Updating of kernel SAD and SPD entries
- Cryptographic: Cookies, Diffie-Hellman, signatures, etc.

In addition, it provides an administrative interface for a command-line tool.

- DNS Library

The DNS library contains a HIT resolver that resolves the HI based on the provided FQDN. If the peer host is a HIP host, it sends the FQDN, HI, and IP address to the HIP daemon.

- Socket API

Maps the corresponding IPv4 compatible LSIs to a HIT for BEET [6] processing.

New socket family has been added, called PF\_HIP, through which the HIP daemon sends the HIP protocol messages.

- BEET mode

The BEET mode is implemented as a integral part of the IPsec. It handles conversions between HITs and IP addresses. BEET mode uses the IDENT extensions to identify the correct SA. The IPsec module handles the decrypting and encrypting of the user data.

## 2.2 Mac OS-X

Doesn't work fully yet. Implementors needed!

## 2.3 Linux

Work in progress.

## 3 Installing and Testing HIP

This section describes shortly the necessary steps to install and test the installed system.

**CHECK CAREFULLY THAT YOU ARE USING THE CORRECT OS VERSION! MOST LIKELY THINGS FAIL AT SOME POINT IF YOU ARE NOT USING EXACTLY THAT VERSION MENTIONED IN THE DOWNLOADED HIP IMPLEMENTATION FILE!**

### 3.1 Installing HIP

The HIP distribution contains a shell script that compiles and installs everything into correct places. It installs the new kernel and replaces the following standard binaries and libraries in the system: setkey, libc, and libipsec. **So consider yourself being warned!**

The installation script is located in the hip/hip4bsd directory. You just need to run command `./install.sh` and follow the instructions on the screen. The script assumes that original FreeBSD sources are located in `/usr/src`. If the directory doesn't exist, the script will ask for the correct location.

When the script finishes you have to reboot your computer to get the new kernel into the memory and to begin to use HIP.

### 3.2 Testing the Installation

After rebooting your machine, you can verify that the HIP works correctly by running the four way handshake against the `woodstock4.hip4inter.net` (if your network is IPv4) or `woodstock6.hip4inter.net` (if your network is IPv6 capable).

Start the hipd daemon `/usr/sbin/hipd`. Run the ping6 command: `ping6 woodstock4.hip4inter.net` or `ping6 woodstock6.hip4inter.net`. With tcpdump you should see four protocol 253 packets going through the interface.<sup>1</sup> Those packets are: I1, R1, I2, and R2. After this base exchange, you should see

---

<sup>1</sup>Protocol number 253 is reserved for experimental use by IANA. This number will change in the future. See RFC3692 [5]

ESP (protocol 50) packets flowing and ping(6) should receive ICMP echo replies from the responder.

Example output from ping6:

```
# ping6 woodstock4.hip4inter.net
PING6(56=40+8+8 bytes) 40e7:2ea8:3bb0:aca0:cbb9:d07d:16df:1378 -->
40ec:c4d7:b9b7:bcdd:6371:82b4:c8db:1672
16 bytes from 40ec:c4d7:b9b7:bcdd:6371:82b4:c8db:1672, icmp_seq=2 hlim=62 time=1.326 ms
16 bytes from 40ec:c4d7:b9b7:bcdd:6371:82b4:c8db:1672, icmp_seq=3 hlim=62 time=1.711 ms
16 bytes from 40ec:c4d7:b9b7:bcdd:6371:82b4:c8db:1672, icmp_seq=4 hlim=62 time=1.242 ms
16 bytes from 40ec:c4d7:b9b7:bcdd:6371:82b4:c8db:1672, icmp_seq=5 hlim=62 time=1.301 ms
16 bytes from 40ec:c4d7:b9b7:bcdd:6371:82b4:c8db:1672, icmp_seq=6 hlim=62 time=1.217 ms
16 bytes from 40ec:c4d7:b9b7:bcdd:6371:82b4:c8db:1672, icmp_seq=7 hlim=62 time=1.273 ms
16 bytes from 40ec:c4d7:b9b7:bcdd:6371:82b4:c8db:1672, icmp_seq=8 hlim=62 time=3.721 ms
16 bytes from 40ec:c4d7:b9b7:bcdd:6371:82b4:c8db:1672, icmp_seq=9 hlim=62 time=1.254 ms
^C
--- woodstock4.hip4inter.net ping6 statistics ---
10 packets transmitted, 8 packets received, 20.0% packet loss
round-trip min/avg/max/std-dev = 1.217/1.631/3.721/0.804 ms

#
```

After you have successfully verified that your installation works properly and you are able to act as an initiator you might want to setup another machine repeating the steps described earlier. With your two HIP machines, A and B, you can test HIP between those machines by setting the HIT and IP of machine A to the hosts file of the machine B. The */etc/hosts* file in machine B should contain lines like:

```
40ec:c4d7:b9b7:bcdd:6371:82b4:c8db:1672 woodstock6
3ffe:800:400:110::133 woodstock6

40ec:c4d7:b9b7:bcdd:6371:82b4:c8db:1672 woodstock4
10.1.1.10 woodstock4
```

If you want you can set your HIT in to the DNS instead of the */etc/hosts* file. Then you should put the HIT into an AAAA record e.g:

```
woodstock4 IN      AAAA   40ec:c4d7:b9b7:bcdd:6371:82b4:c8db:1672
           IN      A      10.1.1.10

woodstock6 IN      AAAA   40ec:c4d7:b9b7:bcdd:6371:82b4:c8db:1672
           IN      AAAA   3ffe:800:400:110::133
```

## 4 Future work

The implementation is updated every now and then. There is no tight schedule for releasing updates.

Please send comments related to the implementation and this document to the following e-mail address:

`hip@hip4inter.net`

## References

- [1] T. Henderson. End-Host Mobility and Multihoming with the Host Identity Protocol draft-ietf-hip-mm-02. Work in progress.
- [2] P. Jokela. Using ESP Transport Format with HIP draft-ietf-hip-esp-00. Work in progress.
- [3] B. Moskowitz. Host Identity Protocol Architecture draft-ietf-hip-arch-03. Work in progress.
- [4] B. Moskowitz. Host Identity Protocol draft-ietf-hip-base-03. Work in progress.
- [5] T. Narten. Assigning Experimental and Testing Numbers Considered Useful. RFC 3692.
- [6] P. Nikander. A Bound End-to-End Tunnel (BEET) mode for ESP draft-nikander-esp-beet-mode-03. Work in progress.