

**Installation and evaluation
of a HIP infrastructure**

Johan Fröjdman

Arcada University of Applied Sciences
Department of Business Administration, Media and Information Technology

Helsinki 2008

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Informationsteknik
Identifikationsnummer:	2299
Författare:	Johan Fröjdman
Arbetets namn:	Installation och utvärdering av en HIP infrastruktur
Handledare:	TkD Göran Pulkkis
Uppdragsgivare:	Arcada
<p>Sammandrag:</p> <p>Nätverksprotokollet Host Identity Protocol, förkortat HIP, är ett experimentellt protokoll som utvecklas av HIP-arbetsgruppen i IETF (Internet Engineering Task Force).</p> <p>HIP protokollet definierar en ny adresseringsnivå ovanför nätverksskiktet i OSI-modellen. Detta protokoll möjliggör identitetsbaserad adressering av noder på Internet. Protokollet förutsätter inga ändringar i existerande Internet-protokoll. Det definierar undersystem för att kringgå begränsningar i den nuvarande dominerande Internet-infrastrukturen.</p> <p>Denna rapport beskriver överskådligt de olika undersystemen i HIP-protokollet och presenterar kommunikations-scenarier för att illustrera protokollets egenskaper. Rapporten presenterar också existerande HIP-realiseringar och deras egenskaper, samt beskriver den HIP-infrastruktur som har installerats i yrkeshögskolan Arcadas datornät i Helsingfors, Finland.</p>	
Nyckelord:	HIP, Host Identity Protocol, DNS, ESP, Mobila datanät, Mobilitet, Multihoming, Internet
Sidantal:	66
Språk:	Engelska
Datum för godkännande:	19.05.2008

DEGREE THESIS	
Arcada	
Degree Programme:	Information Technology
Identification number:	2299
Author:	Johan Fröjdman
Title:	Installation and evaluation of a HIP infrastructure
Supervisor:	Dr. Tech. Göran Pulkkis
Commissioned by:	Arcada University of Applied Sciences
<p>Abstract:</p> <p>The network protocol Host Identity Protocol, abbreviated HIP, is an experimental protocol that is being developed by the HIP Working Group in the Internet Engineering Task Force (IETF).</p> <p>The HIP protocol defines a new addressing layer above the network layer in the OSI model. This protocol enables identity-based addressing of nodes on the Internet. The protocol requires no changes in existing Internet protocols, it defines subsystems with the purpose of bypassing constraints of the currently dominating Internet infrastructure.</p> <p>This report briefly describes the subsystems of the HIP protocol and presents communication scenarios to highlight the features of the protocol. The report also presents existing HIP implementations and their features. Furthermore, the HIP infrastructure that has been installed in the computer network of Arcada University of Applied Sciences in Helsinki, Finland is described.</p>	
Keywords:	HIP, Host Identity Protocol, DNS, ESP, Mobile Networking, Mobility, Multihoming, Internet
Number of pages:	66
Language:	English
Date of acceptance:	19.05.2008

LIST OF TABLES

Table 1: Definition of HIP DNS RR for the Bind configuration file	22
Table 2: ESP suite IDs.....	23
Table 3: Operating systems supported by HIP implementations	27
Table 4: The contents of an automatically generated OpenHIP configuration file hip.conf.....	30
Table 5: The OpenHIP configuration file known_host_identities.xml defines local mapping of HIT/IP.	31
Table 6: The contents of an OpenHIP configuration file my_host_identities.xml.....	32
Table 7: Directory listing of /etc/hip used by InfraHIP.....	34
Table 8: The contents of the default InfraHIP configuration file hipd_config.....	34
Table 9: Directory listing of the source files for the source files of the InfraHIP Java interface (JIP) included in the InfraHIP 1.03 distribution	35
Table 10: A listing of the contents of the /etc/hip directory and the public key	36
Table 11: The contents of a sample /etc/hosts file.....	37
Table 12: Directory listing of the source files used by the Java HIP interface for hip4inter.net.....	37
Table 13: OpenHIP 0.5 / InfraHIP service on crossroads.infracorp.net compatibility test results.....	43
Table 14: Test results of compatibility testing, OpenHIP 0.5 / hip4inter.net services at woodstock4.hip4inter.net and woodstock6.hip4inter.net	43
Table 15: Results of a HIP DNS query on one host in the Arcada HIP system	46

LIST OF FIGURES

Figure 1: Captured IP packets illustrating HIP communication being established	15
Figure 2: An I1 control packet sent between two hosts is captured using Wireshark	16
Figure 3: A R1 packet sent between two hosts is captured using Wireshark	17
Figure 4: An I2 packet sent between two hosts using hip4inter.net is captured using Wireshark.....	18
Figure 5: A R2 packet sent between two hosts is captured using Wireshark	19
Figure 6: A CLOSE packet sent between two hosts using hip4inter.net is captured using Wireshark.....	20
Figure 7: A CLOSE_ACK packet sent between two hosts is captured using Wireshark	21
Figure 8: An ESP packet sent between two hosts doing HIP communication using hip4inter.net, captured using Wireshark	24
Figure 9: The Windows XP application services.msc lists OpenHIP as a service	29
Figure 10: The hipmon program offers a context menu when the HIP service is running	29
Figure 11: The hipmon program listing active HIP associations	32
Figure 12: Structure of the Arcada HIP domain.....	38

LIST OF ABBREVIATIONS

HIP	Host Identity Protocol
DNS	Domain Name System
HIT	Host Identity Tag
HI	Host Identity
RR	Resource Record
SPI	Security Parameters Index
ESP	Encapsulated Security Payload
ORCHID	Overlay Routable Cryptographic Hash Identifiers
API	Application Programming Interface
IP	Internet Protocol
LSI	Local Scope Identifier
ASCII	American Standard Code for Information Interchange
LAN	Local Area Network
WLAN	Wireless Local Area Network
GRPS	General Radio Packet Service
FQDN	Fully Qualified Domain Name

DEFINITIONS AND CONCEPTS

Host

A computer or other device connected to the Internet.

Internet Protocol address

A 32 bit or 128 bit value used to address and contact hosts on the Internet.

Attack

This document refers to an attack as being a hostile attempt to e.g. disturb your network connection (e.g. a Denial of Service (DoS) attack), capture sent data, modify sent data (e.g. man-in-the-middle attack where a host on the route between two computers modify the data before routing it forward).

Cryptographic signature

A cryptographic signature is a numeric value that is used to match data to a certain key in a public key cryptography key pair. It can be used to verify that an e-mail is sent by the person that claims to be the sender or in the case of HIP to sign data to prevent man-in-the-middle attack.

Public key cryptography

In public key cryptography you have two keys, one private and one public. Your private key is used to decrypt and to sign data. The public key can be used to encrypt data and to verify signatures.

Hash

A hash is a numeric value that is calculated from a set of data. The resulting hash is a value of fixed length, usually a few bytes, that attempts to uniquely match the set of data it was calculated from. It can be used to see if data has been modified. If even one bit has changed in the data then the hash should be significantly different.

Host Identity

The Host Identity is a numeric value that provides a globally unique identity. The value must be a RSA or DSA cryptographic key if used for trusted communication and encryption in HIP.

Host Identity Tag (HIT)

A HIT is a 128bit value that is generated from the Host Identity. It consists of a prefix followed by a hash of the Host Identity. It can be used to contact a HIP host on the Internet.

Base64

Base64 is a method of encoding data to a byte stream consisting of only the letters A-Z, a-z, the numbers 0-9, and the characters '+' and '/'. Data encoded in Base64 can be represented as ASCII text files (Josefsson, 2003). The Host Identity is encoded in this format in the HIP RR if it is a numeric key in a public-key cryptography key pair.

Protocol number

The IP protocol packet header contains an integer value representing the protocol number of the protocol embedded in the IP packet payload. The HIP protocol was assigned the number 139 on February 2, 2008 (Internet Assigned Numbers Authority, 2008).

Local Area Network (LAN)

In this document a LAN is a wired Ethernet network that is connected to the Internet.

Wireless Local Area Network (WLAN)

In this document a WLAN is a wireless Ethernet network that is connected to the Internet.

TAP driver

A TAP driver simulates an Ethernet device and works on Ethernet frames on OSI layer 2. Applications can use this virtual device to modify ingoing and outgoing data packets (Krasnyansky, 2000).

TAP is used by some HIP implementations as a solution to LSI mapping.

PREFACE

When I was offered the possibility to do my thesis on the installation of a Host Identity Protocol infrastructure, I thought it would be an interesting opportunity to learn more and take part in the development of new technology.

I would like to thank Göran Pulkkis for giving me the opportunity to participate in this interesting project.

TABLE OF CONTENTS

List of Tables	4
List of Figures.....	5
List of Abbreviations	6
Definitions and Concepts	7
Preface	9
Table of Contents	10
1. Introduction	12
2. Theoretical Background	12
2.1 Scenarios Describing Features of HIP	12
2.2 HIP Overview	15
2.2.1 HIP Control Messaging	15
2.2.2 Using the HIP DNS Extension	21
2.2.3 The Base Exchange Procedure	22
2.2.4 Base Exchange with ESP Protection	22
2.2.5 Rendezvous Servers.....	24
2.2.6 The Native Programming Interface for HIP	25
2.2.7 NAT Traversal.....	25
2.2.8 Mobility and Multihoming	26
2.3 Other Concepts related to HIP	26
2.3.1 Opportunistic Initiation of Communication	26
2.4 Technology related to HIP.....	26
2.4.1 ORCHID	26
2.4.2 OpenDHT	27
3. HIP Implementations	27
3.1 OpenHIP	28
3.2 InfraHIP	33

3.3 HIP4inter.net	35
3.4 PyHIP	37
4. The Arcada HIP Infrastructure	38
5. Conclusions	42
References	47
Appendix A: A HIP communication example.....	51
Appendix B: Patch to bypass configuration parsing errors in the Bind server when Rendezvous Servers are described	55
Appendix C: Overview of the contents of the HIP specifications.....	56
Appendix D: Ideas for a Windows HIP service	60
Appendix E: Internet addresses for HIP related software and information.....	62
Appendix F: List of HIP test servers	64

1. INTRODUCTION

Background and goals

The goal of this thesis work is to install for the Host Identity Protocol a test environment that is available for global use. One part of the project is to evaluate existing HIP implementations and supporting software and install the software that is best suited for this task.

2. THEORETICAL BACKGROUND

The Host Identity Protocol is designed from the need to create a network protocol that adds a new namespace in addition to the two widely used namespaces in the Internet, DNS names and IP addresses. The concept of the protocol is to provide identity based addressing unrelated to the physical location or underlying communication protocols, and to provide identity verification and encrypted communication. This new namespace addresses computers by their identity instead of location as in IP addresses and DNS names. The computers can be assigned several identities or several computers may be addressed by a single identity (distributed computing).

Some of the features of the protocol are primary features of the protocol. Some other features are mechanisms and sub-systems that are needed to realize the concept of the protocol in practice, to work within the constraints of the existing Internet infrastructure.

2.1 Scenarios Describing Features of HIP

There are two network hosts in distinct Internet domains. One host is a desktop computer having a static IP address. The computer is behind a NAT gateway because the Internet address is shared by several computers, and the NAT device is unable to forward incoming connections. The other host is a portable laptop computer that is connected to the Internet through a WLAN, but when a WLAN is unavailable it falls back to GPRS. Both hosts know the host identity of the other host.

Scenario 1a. The mobile computer wants to communicate with the desktop computer.

First the local HIP driver on the mobile computer executes a DNS lookup for a HIP record containing information about the desktop computer's HIT. The DNS lookup responds with a HIP Resource Record containing the IP address, Host Identity and Host Identity Tag of the desktop computer.

The mobile computer then initiates the Base Exchange procedure and sends a HIP control packet I1 to the desktop computer. Because the desktop computer is behind a badly configured NAT device, the packet cannot reach the recipient. The HIP specification describes NAT traversal, but this requires a Rendezvous Server to act in NAT relay mode. The DNS record for the desktop computer does not address a Rendezvous Server.

Scenario 1b. The mobile computer wants to communicate with the desktop computer.

The scenario is set up as the previous scenario, but now the desktop computer has defined a Rendezvous Server in a DNS resource record. Even though it does not need the Rendezvous Server to update its IP address, it needs it for HIP relaying properties of the Rendezvous Server.

Now the mobile computer is able to successfully send an I1 packet through the Rendezvous Server to the desktop computer, by relaying it via the NAT relaying service of a Rendezvous Server. The HIP relay service is a feature offered by the Rendezvous Server. All other control packages are also automatically relayed via this server, and the NAT relay attempts to negotiate a direct communication path between the hosts, so that relaying the ESP packets would not be needed.

The mobile computer knows the HIT of the desktop computer, and thus it fills in the Recipient HIT field in the I1 control packet. It would be possible for the mobile computer to contact the desktop computer also if it only knew the IP address of the desktop computer. This method of contacting is called opportunistic mode (however it does not reflect the central properties and concepts of the Host Identity Protocol).

After the desktop computer has received the I1 packet, other control packets are sent between the desktop and the mobile computer. The control packets establish a trusted connection using the cryptographic properties of the host identities to their advantage. After the control packets are sent and verified by the hosts, a trusted connection is

established. ESP based encryption can be used if it is chosen during the Base Exchange, but it is not mandatory. (Moskowitz & Nikander, 2006)

The HIP connection is now established, and the communicating software can use the HIP interface.

Now we imagine that the mobile computer is moving out of range of the WLAN network, and the physical network connection is lost. It is not in the scope of the HIP protocol to manage the physical connections. Thus it is up to the operating system or third-party software to try to establish an alternative connection. In this case the mobile computer detects that the WLAN connection has been lost and tries to establish an alternative connection. GPRS is the only available method in this case. The mobile computer in this scenario has been set up to automatically choose a network in order of preference and availability.

After the mobile computer establishes the GRPS connection, the HIP system detects that an Internet connection is available and sends to the desktop computer an *UPDATE* message informing about the new IP address and re-establishing ESP security associations. The connection is then re-established without requiring any manual actions by the users of either computer.

When the data communication is to be terminated, the underlying HIP driver transmits *CLOSE* and *CLOSE_ACK* messages which terminate the connection.

Scenario 2. The desktop computer wants to communicate with the mobile computer.

In a similar scenario to the previous one, where only the initiator of the communication has been swapped, the situation is different.

The desktop computer queries the DNS server for HIP information about the mobile computer using the HIT of the mobile computer to perform the query. The query results contain the IP, HI, HIT and in this case the DNS name of a Rendezvous Server. The mobile computer receives the packet and completes the Base Exchange procedure directly with the initiating host, resulting in an established connection.

This differs from the previous case, because the Rendezvous Server only forwards the I1 packet, further communication is done directly between the hosts because the NAT device does not hinder the communication in this case.

2.2 HIP Overview

2.2.1 HIP Control Messaging

The HIP specifications define 8 control packet types, 4 are used to establish a trusted connection and the other 4 packet types are used to manage and maintain an existing connection.

Figure 1 shows network data packets that have been captured and analyzed using the Wireshark network packet analyzer software. Screenshots made using this software are used throughout this report to illustrate the contents of data packets.

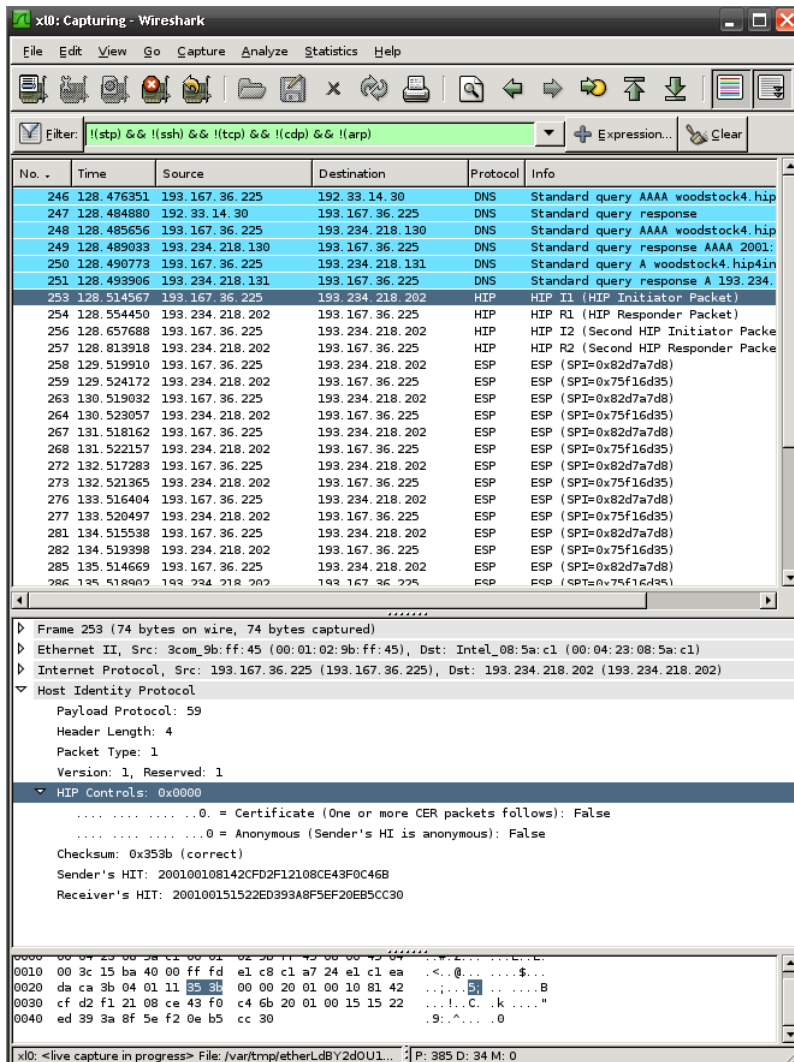


Figure 1: Captured IP packets illustrating HIP communication being established

11 – the HIP Initiator Packet

This packet is sent from the host that initiates the connection to the other party in the communication. The packet contains only the HIT of the initiator and possibly the HIT of the responder. If the HIT of the responder is undefined, then opportunistic mode is used and the responder answers with its HIT (which increases the risk of man-in-the-middle attacks).

In the case that Rendezvous Servers are used, this packet is first sent from the initiator to the Rendezvous Server. The Rendezvous Server checks the IP address of the recipient and then forwards the I1 message to the recipient. Further control messages are sent directly between the communicating hosts.

Figure 2 illustrates the data fields of the I1 packet.

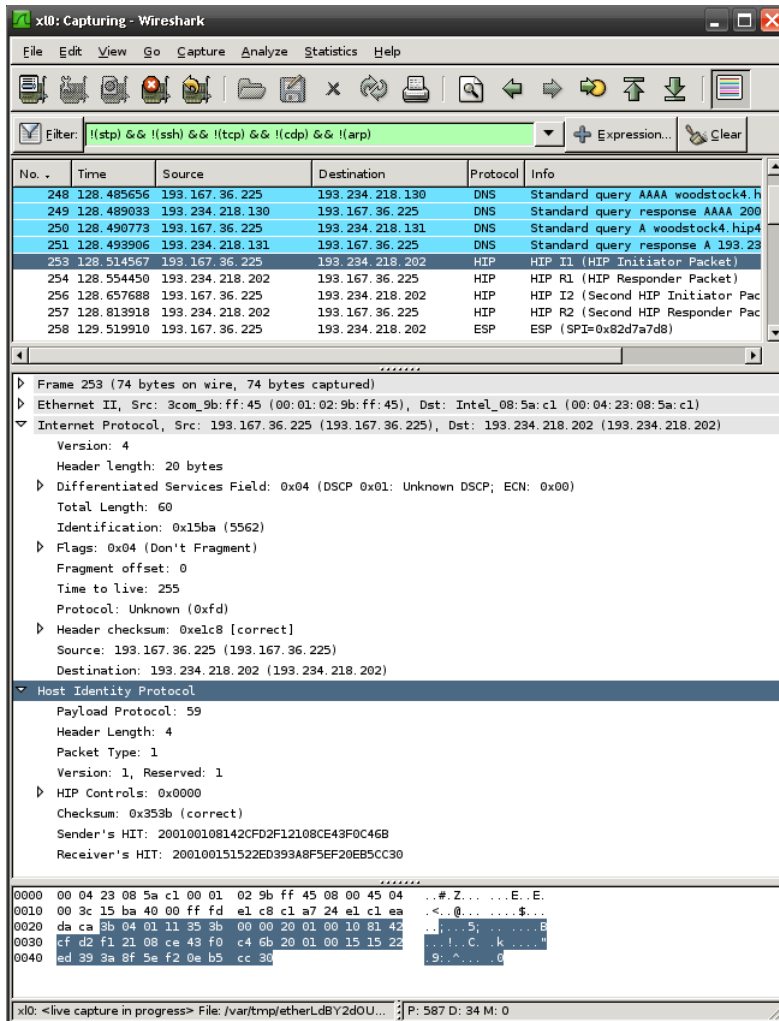


Figure 2: An I1 control packet sent between two hosts is captured using Wireshark

R1 – the HIP responder Packet

The R1 packet is sent from the responder to the initiator in response to the I1 packet, which initiates the Base Exchange. It contains a cryptographic puzzle, a challenge that the initiator must solve. The puzzle contains a random number I, to which the initiator should reply with a number J in the reply packet I2. The number J is a hash of the concatenation of the number I and the HITs of the parties. The I1 packet also contains initial Diffie-Hellman parameters, and a signature of the data. (Moskowitz, Nikander, Jokela, & Henderson, 2008)

Figure 3 illustrates the data fields of the R1 packet.



Figure 3: A R1 packet sent between two hosts is captured using Wireshark

I2 – the Second HIP Initiator Packet

The I2 packet is sent from the initiator to the responder in response to the R1 packet. The packet contains the answer to the cryptographic puzzle sent in the R1 packet. Also included are Diffie-Hellman parameters for the responder. The packet is finally signed before being dispatched.

If the answer to the cryptographic challenge is incorrect then the I2 packet is discarded.

Figure 4 illustrates the data fields of the I2 packet.



Figure 4: An I2 packet sent between two hosts using hip4inter.net is captured using Wireshark

R2 – the Second HIP Responder Packet

The I2 packet is sent from the responder to the initiator in response to the I2 packet. The R2 packet completes the Base Exchange and results in an established connection if all parameters and puzzles are correct at this point.

Figure 5 illustrates the data fields of the R2 packet.

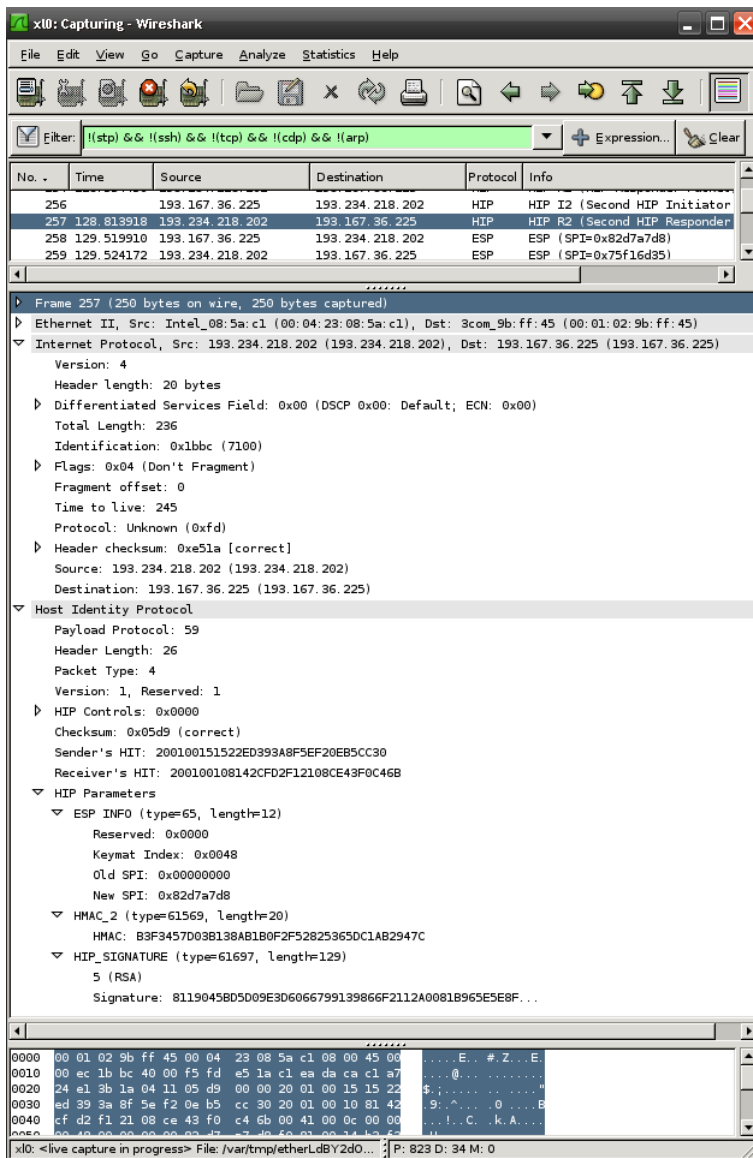


Figure 5: A R2 packet sent between two hosts is captured using Wireshark

UPDATE – the HIP Update Packet

The update packet is used to update information related to a HIP association.

Situations that require this may be updating an existing ESP security association or mobility management (i.e. changed IP addresses).

NOTIFY – the HIP Notify Packet

Notify messages are purely informational message that transmit error messages. They require no actions by the receiving host.

CLOSE – the HIP Association Close Packet

This message is sent to terminate an existing HIP association between two hosts.

Figure 6 illustrates the data fields of the CLOSE packet.

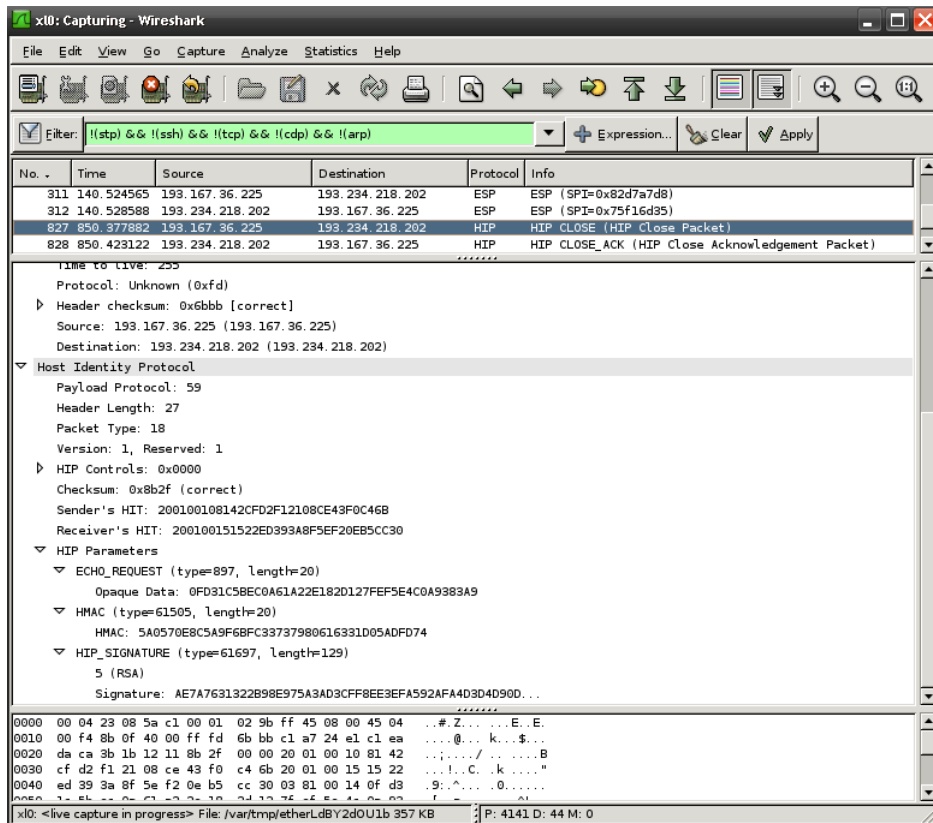


Figure 6: A CLOSE packet sent between two hosts using hip4inter.net is captured using Wireshark

CLOSE_ACK – the HIP Closing Acknowledgement Packet

This message is sent as a response to a CLOSE message, and it has a signature to verify that it is valid. (Moskowitz, Nikander, Jokela, & Henderson, 2008)

Figure 7 illustrates the data fields of the CLOSE_ACK packet.

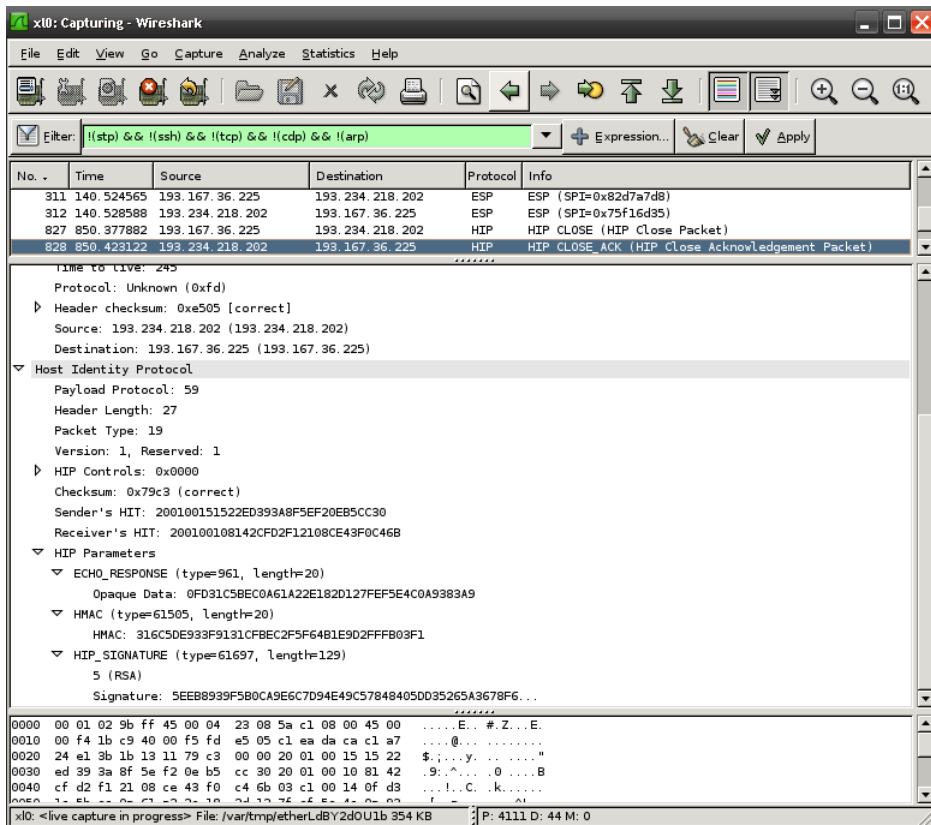


Figure 7: A CLOSE_ACK packet sent between two hosts is captured using Wireshark

2.2.2 Using the HIP DNS Extension

The specifications state that for a host to be reachable to its FQDN it should have the following information stored in the DNS. (Nikander & Laganier, 2008)

- IPv4 and IPv6 addresses are stored in A and AAAA resource records, respectively.
- A HIP resource record contains the host identity (HI), host identity tag (HIT) and IP addresses of possible Rendezvous Servers.
- The HIP resource record thus contains the following information in the given order.
 - o The length of the Host Identity Tag (1 byte)
 - o The algorithm used by the Public Key (1 byte)
 - o The length of the public key (1 byte)
 - o The Host Identity Tag
 - o The Public Key

- An optional list of Rendezvous Servers, listed by DNS name in order of preference and separated by white space

A HIP resource record in a DNS description file has the format shown in Table 1.

Table 1: Definition of HIP DNS RR for the Bind configuration file

IN	HIP	(pk-algorithm
			base16-encoded-hit
			base64-encoded-public-key
			rendezvous-server[1]
			...
			rendezvous-server[n]
)	

The value 1 in the algorithm field indicates that the public key is a RSA key, the value 2 indicates that the public key is a DSA key. (Nikander & Laganier, 2008)

2.2.3 The Base Exchange Procedure

Base Exchange is the procedure where two communicating nodes verify each other's identity and initialize further communication. This is the first procedure in the communication between two nodes.

The Base Exchange consists of sending four data packets between the hosts (Moskowitz, Nikander, Jokela, & Henderson, 2008). The four packet communication is specifically designed to make the protocol resistant to Denial of Service attacks. The communication uses the experimental protocol number 253 or the number 139 that has been defined for the Host Identity Protocol (Internet Assigned Numbers Authority, 2008).

2.2.4 Base Exchange with ESP Protection

The HIP specifications use the HIP protocol with IP protocol number 139 only for transmitting HIP protocol related information and control messages, not for actual data transmission between hosts. Actual data is transmitted using ESP packets (IP protocol number 50). (Kent, 2005)

The Base Exchange that initiates and verifies a HIP connection between two hosts also defines the methods that should be used after the connection has been established. The ESP handshake procedure is performed simultaneously with the Base Exchange, and a successful Base Exchange results in valid security associations that can be used to

encrypt the transmitted data. The methods used for data transmission in ESP packets are standardized in the IPsec specifications, RFC4303. The ESP packet used by HIP has the same format. The ESP packets support encryption of the transmitted data but the data transmitted in an ESP packet is not encrypted if null encryption is defined in ESP. (Jokela, Moskowitz, & Nikander, 2008)

During the Base Exchange, the hosts send ESP related information to each other in order to establish the security associations. The first message, ESP_TRANSFORM, is sent as a parameter of the R1 message of the Base Exchange. This response message defines the ESP transforms that the recipient of the connection is willing to accept. This host sends the I2 HIP control message back to the initiator with attached ESP information containing the chosen transforms and an ESP_INFO parameter containing the SPI value to be used by the host. The initiator responds with the final R2 control packet, which includes the SPI to be used by the recipient. This completes the ESP handshake and the Base Exchange. (Moskowitz, Nikander, Jokela, & Henderson, 2008)

After the initial Base Exchange and the establishment of security associations, both hosts enter the state ESTABLISHED, and the roles of initiator and responder are lost.

The ESP_TRANSFORM message defines from one up to a maximum of six ESP transform suites that the host is willing to use, in order of preference. Table 2 lists the ESP transforms available for use by HIP. The suites ESP-AES-CBC with HMAC-SHA1 and ESP-NUL with HMAC-SHA1 are mandatory to exist in HIP implementations. (Jokela, Moskowitz, & Nikander, 2008)

Table 2: ESP suite IDs

<i>Suite-ID</i>	<i>Value</i>
ESP-AES-CBC with HMAC-SHA1	1
ESP-3DES-CBC with HMAC-SHA1	2
ESP-3DES-CBC with HMAC-MD5	3
ESP-BLOWFISH-CBC with HMAC-SHA1	4
ESP-NUL with HMAC-SHA1	5
ESP-NUL with HMAC-MD5	6

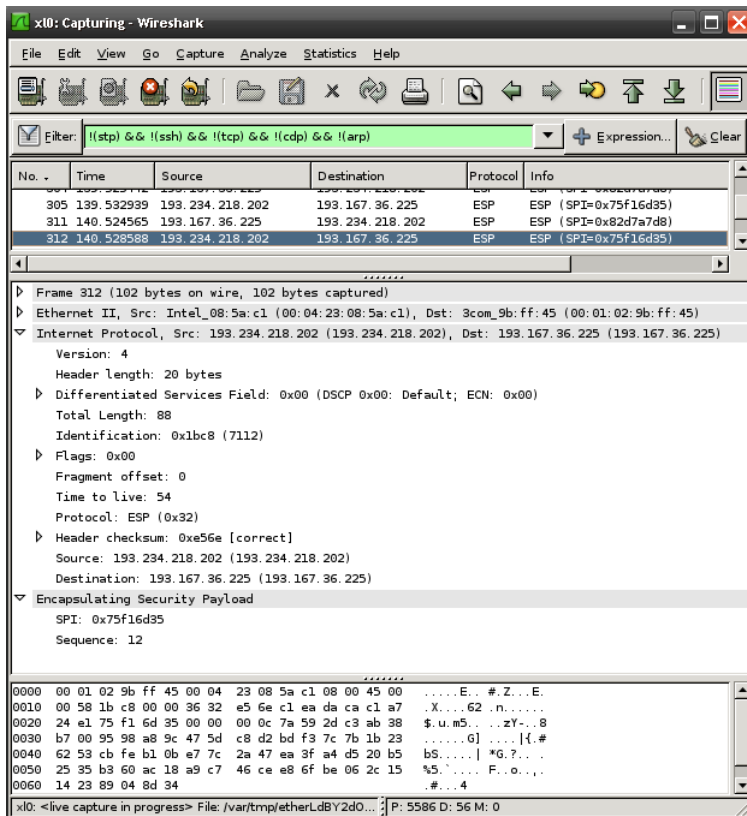


Figure 8: An ESP packet sent between two hosts doing HIP communication using hip4inter.net, captured using Wireshark.

Figure 8 shows an ESP packet that is sent during communication using HIP. The actual traffic being transmitted is a PING command using the ICMP protocol. The traffic is routed through the HIP interface. After the Base Exchange is completed, the actual data traffic is sent through ESP packets.

The Wireshark packet analyzer can only decipher non-encrypted information from an ESP packet, which is the SPI that identifies the security association, and a packet sequence number. In the case that the NULL transform is used, the data packet is not encrypted and plain-text messages can be easily decoded from the ESP packet.

2.2.5 Rendezvous Servers

Rendezvous Servers are used as a solution to constraints of the existing Internet infrastructure. One of the primary requirements of a protocol is that hosts should be accessible just by providing the identity of the host. The traditional Internet infrastructure with IP addresses and DNS names does not provide a solution to this.

Thus the solution used by HIP is a Rendezvous Server, as a bridge between locators (IP addresses) and identities (HIT and HI keys).

The Rendezvous Server is used when a host is initiating a connection to another host, after a DNS lookup has returned the IP number of a Rendezvous Server instead of the IP number of a host. (Laganier & Eggert, 2008)

Usage of Rendezvous Servers requires registration to the rendezvous service using the registration method defined in the HIP specifications (Laganier, Koponen, & Eggert, 2008).

2.2.6 The Native Programming Interface for HIP

Network programming using HIP native structures and functions is realized as an extension to the existing sockets programming interface. A new address family AF_HIP is defined and is used to indicate a HIP host in the programming structures and calls.

The HIP address description data structure contains the length of the structure, the family (AF_HIP), the receiving port, flags and the Host Identity Tag. (Komu & Henderson, Basic Socket Interface Extensions for Host Identity Protocol (HIP), IETF draft, 2008)

Generally HIP specific programming is analogue to standard sockets programming replacing the IP concepts with HIP counterparts.

2.2.7 NAT Traversal

The HIP specifications define a workaround to the problem of NAT devices not correctly forwarding incoming HIP packets sent to a HIP host. The solution is that the host or hosts having problems receiving packets establish an outgoing connection to a Rendezvous Server that helps to successfully relay messages to the recipient. The reason why this works is that the NAT device does not control outgoing connections.

HIP relaying is a service of the Rendezvous Server.

When a transmission channel has been established through a Rendezvous Server, the hosts can negotiate the method used to transmit the ESP packets. (Komu, et al., 2008)

2.2.8 Mobility and Multihoming

One of the basic features of HIP is the decoupling of the locator and the identity. This opens up possibilities for mobile devices to change the locator (i.e. physical network connection and IP address) on-the-fly while retaining the same identity. The mobility and multihoming specifications define the how HIP should handle sudden IP address changes.

When the IP address changes, the physical connection is lost, and the host that got a new IP address sends an UPDATE packet to the other host to inform it of the updated IP address. Then the Security Associations for the ESP transmission is renewed.

(Nikander, Henderson, Vogt, & Arkko, 2008)

It is not yet defined in the specifications what to do if both hosts change their IP address simultaneously in a way that neither UPDATE packet is received by the intended recipient.

2.3 Other Concepts related to HIP

2.3.1 Opportunistic Initiation of Communication

Opportunistic mode refers to the initiation of communication without knowing the identity of the host, with which the communication is initiated. Then an initiator packet is sent to the recipient without any identity information concerning the recipient. This practice increases the risk of man-in-the-middle attacks. It is recommended to obtain identity information about the other party by performing a DNS lookup for a HIP RR.

(Moskowitz, Nikander, Jokela, & Henderson, 2008)

2.4 Technology related to HIP

2.4.1 ORCHID

The specifications for the Overlay Routable Cryptographic Hash Identifiers (ORCHID) define a class of addresses having a form identical to IP addresses. These addresses are used for a different purpose than normal IP addresses, since they are used as globally

unique identifiers. They are created using a hash of the data and identified by a pre-selected prefix, resulting in 128-bit values that have an identical construction as IPv6 addresses (Nikander, Laganier, & Dupont, An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers (ORCHID), IETF RFC 4843, 2007). The prefix is used to distinguish an ORCHID from ordinary IP addresses. (Blanchet, 2008)

2.4.2 OpenDHT

InfraHIP and OpenHIP use the OpenDHT system as a method of mapping IP and HIT information to each other. OpenDHT is a distributed hash table, enabling storing and retrieving small amounts of data on the Internet using a numerical key as the handle to the data. As of January 2008, the OpenDHT network consists of 150 nodes. By using the Distributed Hash Table system, easy mapping of IP and HIT information is possible. OpenDHT is not officially included in the HIP specifications but it is used as a solution to the problem of finding other HIP hosts when DNS is insufficient.

The Internet draft document HIP DHT Interface describes benefits and methods of using DHT instead of DNS in getting HIP/IP mappings. DHT is considered faster and more suitable for mapping IP and Host Identities than DNS. (Ahrenholz J. , 2008)

3. HIP IMPLEMENTATIONS

Present HIP implementations consist of three competing software packages, InfraHIP by the Helsinki Institute of Information Technology, OpenHIP by the Boeing Company, and hip4inter.net by Oy LM Ericsson Ab. An implementation using Python also exists, PyHIP. Table 3 shown operating system support for all HIP implementations as well as eventual constrains that apply.

Table 3: Operating systems supported by HIP implementations

<i>Distribution</i>	<i>Windows XP</i>	<i>Linux 2.4</i>	<i>Linux 2.6</i>	<i>FreeBSD 6</i>	<i>MacOS X</i>
OpenHIP 0.5	Yes, no Rendezvous Server	Yes, only user mode	Yes	-	Yes, no Rendezvous Server
InfraHIP 1.03	-	-	Yes	-	-
hip4inter.net 2008.04.25	-	-	-	Yes	-
PyHIP 2003-11-15	-	Yes	Unknown	-	-

3.1 OpenHIP

OpenHIP is an open source HIP implementation maintained by the Boeing Company. The homepage of the project is www.openhip.org. The implementation has Windows XP, MacOS X and Linux versions. The implementations for MacOS X and Windows exist entirely in user-space with no code inside the operating system kernel. The Linux version has both user-space and kernel mode versions, but the kernel mode version is lagging behind in development. Being an open source project, it uses some existing open-source components to do some of the work. It uses OpenSSL for cryptographic functions and LibXML for XML support in the configuration files. (Boeing Company, 2007)

The configuration of the local host identity and known host identities uses XML configuration files.

The OpenHIP implementation running in user space relies on a TAP driver. OpenHIP for Windows and MacOS X support initiation of connections using LSIs or HITs only through a TAP device. The TAP device is a virtual device running entirely in software.

TAP is used to create a network bridge. In practice it works by passing packets received from a certain address associated with an existing HIP transaction to a LSI address. The association between the LSI and the actual IP address is statically defined in the known hosts' database, but OpenHIP also supports fetching HIP data from DNS. (Boeing Company, 2007)

The Linux version supports kernel mode by patching the 2.6 version of the Linux kernel. Running in kernel mode reportedly increases the performance of the software.

The Linux version also contains a built-in Rendezvous Server.

Installation

Installing OpenHIP requires that the IPv6 component of the operating system is installed. While IPv6 is not directly used by OpenHIP, it is used in the generation of the host identity.

The installation itself is trivial and is done using a graphical installation program.

Using OpenHIP on Windows requires that the IPsec service is not running, because then two services would listen to ESP packets. The Windows IPsec service is automatically started by default, but it can be configured not to start automatically. (Boeing Company, 2007)

Graphical walkthrough of the use of the Windows version of OpenHIP

The HIP service can be started through the Windows service manager (Figure 9) or through the application *hipmon* that installs an icon into the Windows task bar. The *hipmon* application offers a menu with shortcuts to common operations related to the HIP software.



Figure 9: The Windows XP application services.msc lists OpenHIP as a service

The icon that is installed into the Windows task bar when *hipmon* is running can be used to control and configure OpenHIP (Figure 10).

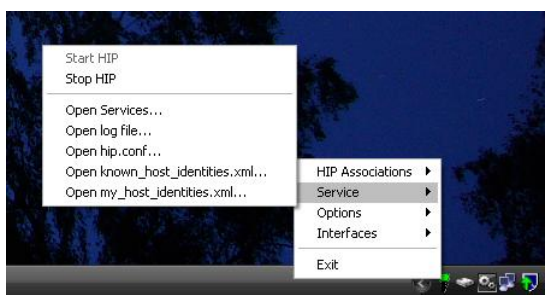


Figure 10: The hipmon program offers a context menu when the HIP service is running

The file *hip.conf* is automatically generated by HIP and offers low level configuration options for the HIP service (Table 4).

Table 4: The contents of an automatically generated OpenHIP configuration file hip.conf

```
<?xml version="1.0" encoding="UTF-8"?>
<hip_configuration>
  <min_lifetime>96</min_lifetime>
  <max_lifetime>255</max_lifetime>
  <reg_type_rvs>1</reg_type_rvs>
  <lifetime>217</lifetime>
  <reg_type>1</reg_type>
  <cookie_difficulty>10</cookie_difficulty>
  <packet_timeout>10</packet_timeout>
  <max_retries>5</max_retries>
  <sa_lifetime>900</sa_lifetime>
  <send_hi_name>yes</send_hi_name>
  <dh_group>3</dh_group>
  <dh_lifetime>900</dh_lifetime>
  <r1_lifetime>300</r1_lifetime>
  <failure_timeout>50</failure_timeout>
  <mssl>5</mssl>
  <ual>600</ual>
  <hip_sa>
    <transforms>
      <id>1</id>
      <id>2</id>
      <id>3</id>
      <id>4</id>
      <id>5</id>
      <id>6</id>
    </transforms>
  </hip_sa>
  <esp_sa>
    <transforms>
      <id>1</id>
      <id>2</id>
      <id>3</id>
      <id>4</id>
      <id>5</id>
      <id>6</id>
    </transforms>
  </esp_sa>
  <disable_dns_lookups>no</disable_dns_lookups>
  <disable_notify>no</disable_notify>
  <disable_dns_thread>no</disable_dns_thread>
  <enable_broadcast>no</enable_broadcast>
</hip_configuration>
```

The file *known_host_identities.xml* (Table 5) defines a local mapping of HIT/IP information, as seen in Table 5. It can be used when HIP information is not available in the DNS HIP record of a host. Each *<host_identity>* tag contains information about a remote host, the DNS name, the HIT, the IP address and the LSI. If the host is registered to a Rendezvous Server, a *<rvs>* tag could be used to address the Rendezvous Server. (Boeing Company, 2007)

Table 5: The OpenHIP configuration file *known_host_identities.xml* defines local mapping of HIT/IP.

```
<?xml version="1.0" encoding="UTF-8"?>
<known_host_identities>
  <host_identity alg="RSA" alg_id="5" length="128" anon="no" incoming="yes">
    <name>hipserver.mct.phantomworks.org-1024</name>
    <HIT>200100144dcd2a09074acaee02a0ec4a</HIT>
    <addr>130.76.43.74</addr>
    <LSI>1.160.236.74</LSI>
  </host_identity>
  <host_identity alg="RSA" alg_id="5" length="128" anon="no" incoming="yes">
    <name>hipproxy-1024</name>
    <HIT>406E4B13CA8BEEDEF084A4F3538DB2D2</HIT>
    <addr>192.76.227.36</addr>
    <LSI>1.1.0.1</LSI>
  </host_identity>
  <host_identity alg="RSA" alg_id="5" length="128" anon="no" incoming="yes">
    <name>crossroads.infracorp.net-1024</name>
    <HIT>20010019b6738406e32d6754db0bcde7</HIT>
    <addr>193.167.187.134</addr>
    <LSI>1.13.205.231</LSI>
  </host_identity>
  <host_identity alg="RSA" alg_id="5" length="128" anon="no" incoming="yes">
    <name>hip.arcada.fi-1024</name>
    <HIT>200100708142cfd2f12108ce43f0c46b</HIT>
    <addr>193.167.36.225</addr>
    <LSI>1.240.196.107</LSI>
  </host_identity>
  <host_identity alg="RSA" alg_id="5" length="128" anon="no" incoming="yes">
    <name>xray.hip.arcada.fi-1024</name>
    <HIT>2001001ba9b870bd79e76ff8fc59fcb6</HIT>
    <addr>193.167.36.226</addr>
    <LSI>1.89.252.182</LSI>
  </host_identity>
</known_host_identities>
```

The contents of the OpenHIP configuration file *my_host_identities.xml* (Table 6) define the local HIP nodes identity; an example of this file is seen in Table 6. The HIT is calculated based on this information. The contents of this file should only be shared if

the identity is intended to be shared. Making the contents publicly known would compromise the HIP identity. (Boeing Company, 2007)

Table 6: The contents of an OpenHIP configuration file my_host_identities.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<my_host_identities>
  <host_identity alg="RSA" alg_id="5" length="128" anon="no" incoming="yes"
rlcount="10">
    <name>johans_pc-1024</name>

<N>DE4D0FE078D7059FD5B91EC825ACFD4067752F83F62938C5686CC5A349AF81B86E4AEB54BC749A91DCB99
B7029E55DED6B442050926CC0E1D3E96AB670B140CB1EEDE0C816608AA5F8486AE0344DBDBBB8844AF7033C0
00A2BA652D25AC33ABCCD30F92B644F0E24F0CFCE19BD546A519F1B3AC09208A49C4C3CBCBE2974660B</N>
    <E>010001</E>

<D>501167978D4EBD3ADDAE8B864419391FFF18864BA640C8BA5DA0FB18997CA5C5875699FC2A5290B427A2
3CF5977C0C603EDACC9F8836D51A9971815902F77CB454BF02A26311214E41B5954B5226810BF395C1402F11
96B5C90C822B203670A94D6B1A98E84BFFDD7BB88BF7AD440CA457682D0ADB45DE14BA92E40D821E499</D>

<P>FB2FDCEAACED52E9E11933B3C87631BBB2879DD01586C6D78C9262B6B1AFB16ACFF4DEC03450D5CE922BD
82F21C7A6BEEFEF32F3F6043111FD276FD29EDDCBCF</P>

<Q>E28F818C6A98E6D48B5AED781D8CAE39CADBA911FC7FDF468884B589FC953E7209B6EE178116FEE7E49E1
3CB717672E11A55DB738B1933DF3B43911D3ACDA505</Q>

<dmp1>632931FD32E73DA436C8CC305D22CBDF5D4B4C71A90DDBD19C1CEDFB518A1A7020487AE745BFE3A7F8
EC761BA52A3C19847ED98E95C071DEB821A9736C6402E1</dmp1>

<dmq1>499968F25EA6B14616C74121A627CF6982D6FD0394CD4D9132443A5D4B2A8890AF4B37E976CC2B9ED5
8BECCE485BBC521826101666F796FC4BDA5547F3945EA9</dmq1>

<iqmp>AC62AF5958E648D057029AA5F440EBD92C4D65C74B286CB5ADFD5BF620D30F71539E3D39DE5D85061B
B86BC4ECB5EA8989AA095C403049EA784E53F5B24FB1FD</iqmp>
    <HIT>2001:17:c0ac:5e87:deaa:2b0:5b1e:817</HIT>
    <LSI>1.30.8.23</LSI>
  </host_identity>
</my_host_identities>
```

After a HIP session has been established, the HIP association is listed in the context menu (Figure 11).

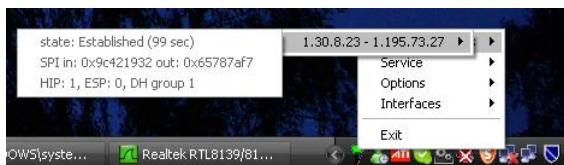


Figure 11: The hipmon program listing active HIP associations

Other HIP related software by the Boeing Company

The OpenHIP project homepage also hosts other projects related to HIP. Two noteworthy projects are the DNS extensions and Wireshark extensions.

The DNS extensions are used by patching the sources of the standard Bind source code.

The Wireshark extensions enable HIP packets other than ESP packets to be analyzed by the packet-analyzer program Wireshark.

The Wireshark screenshots in this document are created by the Wireshark program with the HIP patch.

3.2 InfraHIP

The InfraHIP project (formerly HIPL) is an open source project hosted by the Helsinki Institute of Information Technology. It is only available to Linux 2.6 compatible operating systems. The project homepage is www.infrachip.org.

The Linux kernel must be compiled with a set of options specified in the project documentation. The Linux distribution Slackware version 12.0 compiles the required kernel options by default (HIIT, 2008). This was verified by installing the operating system and running the *make xconfig* utility.

The presently latest version as of March 2008, *hipl--main--2.6--patch-317*, uses the recently changed standardized protocol number 139 for HIP instead of the general experimentation number 253 (HIIT, 2008).

Installation

Installation of InfraHIP is based on the traditional UNIX method, by running configuration and compilation scripts (HIIT, 2008).

The software requires certain options to be built into the Linux kernel. These options offer the user-mode application the capabilities needed to work as a HIP service.

Running the daemon consists of running the program */usr/sbin/hipd*. It can be defined to start automatically upon start-up. The method to do this depends on the Linux distribution used (e.g. in Slackware it can be done by a script in the directory */etc/rc.d*).

Configuration

The directory `/etc/hip` contains the configuration files for InfraHIP (Table 7).

Table 7: Directory listing of `/etc/hip` used by InfraHIP

```
hip_host_dsa_key_anon
hip_host_dsa_key_anon.pub
hip_host_dsa_key_pub
hip_host_dsa_key_pub.pub
hip_host_rsa_key_anon
hip_host_rsa_key_anon.pub
hip_host_rsa_key_pub
hip_host_rsa_key_pub.pub
hipd_config
hosts
```

The file `hipd_config` is used to define static HIP/IP mappings, to enable the HIP service to automatically use rendezvous mode, and to enable NAT mode (Table 8).

Table 8: The contents of the default InfraHIP configuration file `hipd_config`

```
# Format of this file is as with hipconf, but without hipconf
prefix.
# add map HIT IP      # preload some HIT-to-IP mappings to hipd
# add service rvs     # the host acts as HIP rendezvous
# nat on              # the host is behind a NAT
# debug none         # no debugging messages will be displayed
```

Rendezvous mode

The rendezvous service can be enabled by running the HIP configuration tool `hipconf`. In InfraHIP the command to enable the HIP rendezvous daemon is `hipconf add service rvs`.

If the host that wants to use the rendezvous service also is running InfraHIP, then it can start using the rendezvous service by executing the command `hipconf add rvs <RVS-HIT> <RVS-IP>`.

Other included software components

The InfraHIP 1.03 distribution also contains a JIP API, a graphical configuration tool and several programs for testing the communication.

JIP API

This is a Java programming API that enables HIP enabled Java. It is divided into two parts, a simple API that only defines that all HIP traffic generated by the program is routed through HIP, and an API that can be used to create HIP socket connections.

The documentation file `/jip/package.html` inside the InfraHIP 1.03 distribution describes the use of the JIP, and recommends the HIP sockets over the global API if access the program uses socket programming. Table 9 lists all the files in the JIP API.

Table 9: Directory listing of the source files for the source files of the InfraHIP Java interface (JIP) included in the InfraHIP 1.03 distribution

```
HipAddress.java
HipServerSocket.java
HipServerSocketFactory.java
HipSocket.java
HipSocketFactory.java
HipSocketImpl.java
HipSocketImplFactory.java
Makefile
NativeInputStream.java
NativeOutputStream.java
hip-address.c
hip-socket-impl.c
hip-socket.c
native-stream.c
package-list
package.html
```

Graphical configuration tool

The InfraHIP 1.03 distribution contains a tool that can be used to graphically configure InfraHIP through an X11 window. The program name is *hipagent*.

3.3 HIP4inter.net

The hip4inter.net project (<http://www.hip4inter.net>) is an open-source project hosted and maintained by Nomadic Labs at Oy LM Ericsson Ab. It currently supports only FreeBSD version 6.1 (as of May 2008).

It is the leanest of the three major implementations. It contains basic Base Exchange and communication abilities but currently no Rendezvous Server. The latest version as of May 2008, release 2008.04.25, officially supports RFC 5201 (Host Identity Protocol), RFC 5202 (Using the ESP Transport Format with HIP), RFC 5206 (End-Host Mobility

and Multi-homing) and draft-nikander-esp-beet-mode-08 (A Bound End-to-End Tunnel (BEET) mode for ESP) (Oy LM Ericsson Ab, 2008).

A version of hip4inter.net with rendezvous server capability is available to participants of the WISEciti project as of May 15, 2008.

The software consists partly of a server application and partly of HIP code inserted into the operating system kernel. (Oy LM Ericsson Ab, 2005)

When a user tries to establish a connection using a Host Identity Tag, the HIP service takes over and manages the connection. If a machine name is used, it is resolved to a HIT using AAAA records in the DNS database or in the local */etc/hosts* file. If the HIT cannot be resolved, the connection cannot be established a “no route to host” situation occurs. The HIP service uses ORCHID prefixes to distinguish a HIT from a real IPv6 address.

Installation

An automated installer performs the entire installation process, including recompilation of the kernel. The HIP service, i.e. daemon, can be started by running it manually (by starting the program */usr/sbin/hipd* as the root user), or it can be configured to start automatically upon start-up by adding the entry *hipd_enable="YES"* to the */etc/rc.conf* file (Oy LM Ericsson Ab, 2005).

Configuration

The HIP software does not require any configuration to be used. It automatically generates a HIP identity into the directory */etc/hip* (Table 10).

Table 10: A listing of the contents of the */etc/hip* directory and the public key

```
-bash-2.05b$ ls
2001:70:8142:cf2d:f121:8ce:43f0:c46b.priv
2001:70:8142:cf2d:f121:8ce:43f0:c46b.pub
hipkey.priv
hipkey.pub

-bash-2.05b$ cat hipkey.pub
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC45W/XQKNpd03NeoANzcY6qGTe
pPGco1Vjb5SGpSXPCwDazkdA5XC3m+VecFz59riIrGPuG91TnCl7xStQ9c03DZGJ
aEOiiwSeMys2t/mOv9BrCvaSKwqyg+tzxtIbdjlnExqYQMr+9eCgHdaJCYgcAdOT
y7rqyuJwG4AZrpZ8owIDAQAB
-----END PUBLIC KEY-----
```

To the file `/etc/hosts` can be added a HIP/IP mapping that is used for local DNS lookup (Table 11).

Table 11: The contents of a sample `/etc/hosts` file

```
402A:913E:2B14:BD8E:6820:6ABE:9DE6:78C8 hipserver.mct.phantomworks.org
130.76.43.74 hipserver.mct.phantomworks.org

402A:913E:2B14:BD8E:6820:6ABE:9DE6:78C8 hipserver.mct.phantomworks.org
130.76.43.74 hipserver.mct.phantomworks.org
```

Usage

Applications utilizing HIP can be developed using the HIP socket API.

Connection test can be done e.g. using the `ping6` program that is installed with the distribution. (Oy LM Ericsson Ab, 2005)

Rendezvous service

This publicly available distribution does not presently include a Rendezvous Server. A version of hip4inter.net with Rendezvous Server capability is available to participants of the WISEciti project as of May 15, 2008.

HIP interface for Java

Hip4inter.net contains a sparsely documented Java interface for HIP. It is included in the main software distribution package. Table 12 lists the source for the interface.

Table 12: Directory listing of the source files used by the Java HIP interface for hip4inter.net

```
-bash-2.05b$ pwd
/home/johan/hip/hip4bsd/src/hip/jhipe/jhipe
-bash-2.05b$ ls
HIPInterfaceImpl.java
-bash-2.05b$ ls ..
HIPInterfaceImpl.c      Makefile          jhipe
HIPTest.java           README           org
-bash-2.05b$
```

3.4 PyHIP

Andrew McGregor has made an implementation of HIP using the Python programming language called PyHIP. It was last updated in November 2003. (Boeing Company, 2007). A download link is available in Appendix E.

The readme file included with the PyHIP distribution describes the usage of the software as well as software- and operating system requirements.

No connection tests using this implementation were done for this thesis.

4. THE ARCADA HIP INFRASTRUCTURE

The Arcada HIP infrastructure, as of May 2008, is made up of three physical computers (Figure 12).

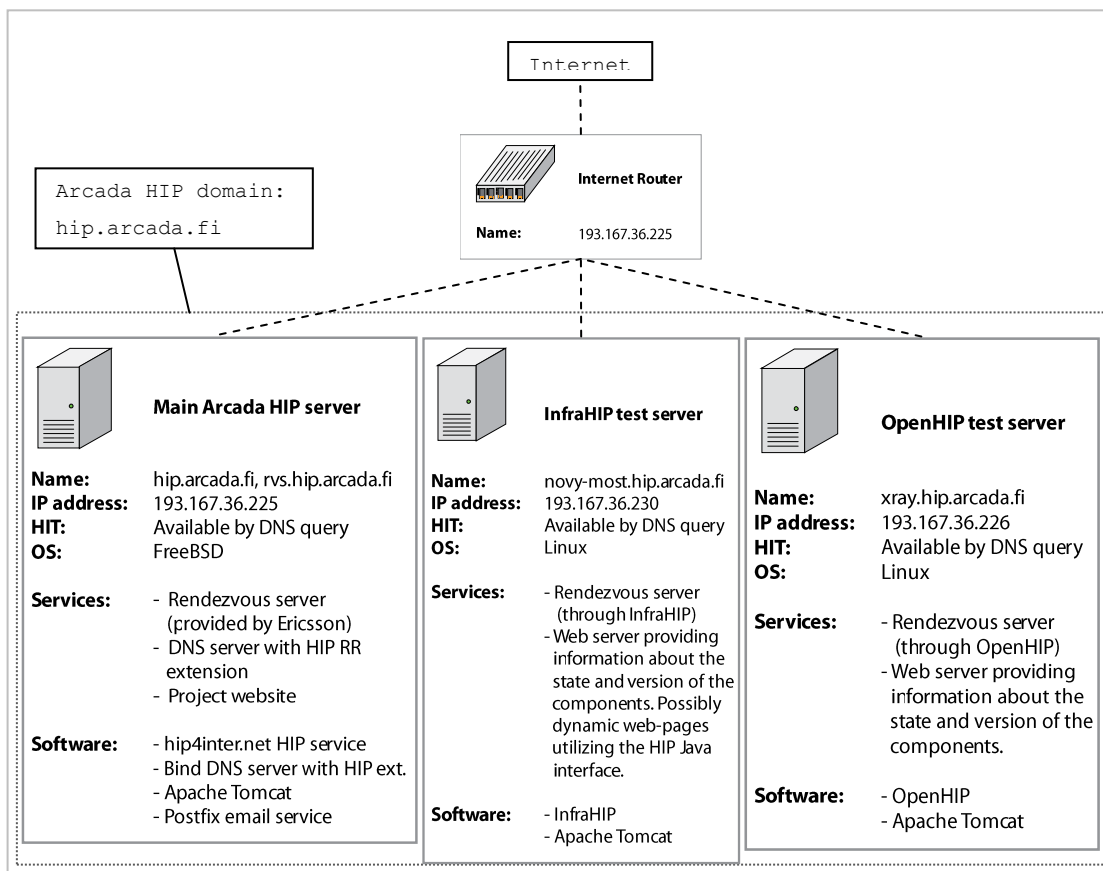


Figure 12: Structure of the Arcada HIP domain

Main HIP server

The operating system used is FreeBSD 6.1. This is the operating system required for current versions of hip4inter.net as of May 2008. The kernel is automatically patched with HIP related functionality when this HIP implementation is installed.

The HIP implementation currently used is HIP4inter.net version 2007.06.11.

The DNS service is a Bind server that has been patched with the HIP DNS extensions provided by the Boeing Company through the OpenHIP project homepage. The Bind server was further patched to enable the use of Rendezvous Servers in the resource record (see appendix A).

The Web server is Tomcat. Tomcat was chosen because there is support for the Java platform through both HIP4inter.net and InfraHIP implementations. These distributions contain Java interfaces for HIP. Thus web-applications using HIP can be developed, which can be valuable if used as a learning platform in an educational environment. The server can provide real-time status about the state of the HIP system, provide eventual services using e.g. the HIP Java API and show the versions of the software used.

Test server with InfraHIP

The operating system used is Slackware 12.0 which has the Linux kernel version 2.6. All the kernel options required by InfraHIP are included in the kernel that is installed by default, this is not stated anywhere but it was verified after checking what kernel options Slackware 12.0 install by default.

The HIP software used is the presently newest version of InfraHIP, version 1.0.3.

This machine can act as a Rendezvous Server without installing any additional software, as the Rendezvous Server ability is included with InfraHIP.

The server has a Tomcat web server installed to provide real-time status about the state of the HIP system, to provide eventual services using e.g. the HIP Java API, and to show the versions of the software used.

Test server with OpenHIP

The operating system used is currently Linux with kernel version 2.6.13-5. The reason for using this outdated version is that the latest version of the OpenHIP kernel patch is for this version, dated May 17, 2006. (SourceForge, 2008)

The HIP software that is installed is OpenHIP version 0.5 for Linux.

This machine can act as a Rendezvous Server without installing any additional software, as the Rendezvous Server ability is included with OpenHIP. The Rendezvous Server ability is only included in the Linux version of OpenHIP (Boeing Company, 2007).

The server has a Tomcat web server installed to provide real-time status about the state of the HIP system, to provide eventual services using e.g. the HIP Java API and to show the versions of the software used.

Proposition for a HIP system running on a virtual server

While it is possible to run the test servers on dedicated machines for each HIP implementation and version, as it is done as of May 2008, this has some caveats. Because the HIP implementations are regularly updated, it is not practical to re-install the operating system each time a new HIP implementation is to be installed.

A more practical solution would be one single powerful server that runs virtualization software. Services that are available on the host operating system would not have to run on the virtual operating systems. The software that doesn't have to run on virtual servers depends on the host operating system. By using virtual hard drive images instead of partitions, it is more convenient to backup, manage and restore system snapshots.

A Linux 2.6 host system with Xen as the virtualization software would be a cost effective solution. There is step-by-step guide in (FreeBSD, 2008) over how to install FreeBSD as a guest operating system on a Linux system running Xen. The webpage does not mention which versions of FreeBSD are supported. Some discussion forums on the Internet mention problems associated with running FreeBSD through Xen (e.g. http://groups.google.com/group/yuanjue/browse_thread/thread/ec4261c9511c1dca, retrieved May 5, 2008). (FreeBSD, 2008)

An alternative solution would be to use commercial virtualization software that officially supports FreeBSD. There are a few candidates:

- Parallels Desktop 3.0 for Mac (Parallels, 2008)
- VMware Fusion (for Macintosh) (VMware, 2008)

These products officially support FreeBSD 6.1 as well as Linux 2.6 systems. These products are marketed as virtualization products for desktop computers. (VMware, 2008) (Parallels, 2008).

VMware Fusion has the advantage that it supports running several virtual machines simultaneously, and it supports multiple processors which would be an advantage on a Mac Pro (VMware, 2007).

Another important requirement is the availability of virtual network devices. If several implementations of the HIP protocol are available simultaneously, then their services should be behind dedicated IP addresses, one for each implementation. VMware Fusion supports up to ten virtual network devices (VMware, 2007), while Parallel Desktops supports five virtual network devices (Parallels, 2008).

A candidate for the server hardware would be Mac Pro, which has eight processor cores as of May 2008 and support for up to 32GB RAM. (Apple Inc., 2008).

5. CONCLUSIONS

The state of the HIP protocol and supporting implementations

As of May 5, 2008, there are three major HIP implementations, OpenHIP by the Boeing Company, hip4inter.net by Oy LM Ericsson Ab and InfraHIP by Helsinki Institute of Information Technology.

It is clearly noticed that the developers of the HIP implementations concentrate primarily on protocol level issues and less on application compatibility. OpenHIP offers no native programming API for HIP; it relies solely on “legacy” mode, i.e. the use of Local Scope Identifiers. InfraHIP does offer native NIP network programming through the HIP socket address family AF_HIP. This feature is added automatically upon install as the Socket functions and structures of the operating system are patched and recompiled.

It is clear that the HIP implementations should be run using virtualization software. Then many implementations can be executed on one server. Managing and updating the distributions would also be significantly more effective.

Mobility

Mobility was tested using two OpenHIP nodes as documented in Appendix A. Changing network attachment between LAN and WLAN was successful, since the connection did not break. Changing from LAN to Sonera GPRS was unsuccessful even though the GPRS connection was established. Mobility tests with OpenHIP, InfraHIP and hip4inter.net were all successful as reported by Jeffrey Ahrenholz, as discussed in (Ahrenholz, 2007).

Compatibility

Jeffrey Ahrenholz from Boeing reported on compatibility tests between OpenHIP 0.5 and InfraHIP and hip4inter.net, in (Ahrenholz, 2007).

Table 13 and

Table 14 present the results from that article. As the article does not specify when the tests were carried out and what version of InfraHIP that the server crossroads.infrachip.net was running, it can only be assumed that the server used the

most recent version of that time, InfraHIP 1.02. The version of hip4inter.net is not mentioned either.

Table 13: OpenHIP 0.5 / InfraHIP service on crossroads.infracorp.net compatibility test results

<i>Test scenario</i>	<i>Results</i>
Base Exchange (IPv4)	Successful
Base Exchange (IPv6)	Successful
IPv4 readdress OK from wired to wireless, reportedly “after updating OpenHIP code (see the CVS version)” (Ahrenholz, 2007)	Successful
CLOSE/CLOSE ACK	Successful
Accessing Web service at crossroads.infracorp.net through HIP and IPv4	Successful
Accessing Web service at crossroads.infracorp.net through HIP and IPv6	Successful

Table 14: Test results of compatibility testing, OpenHIP 0.5 / hip4inter.net services at woodstock4.hip4inter.net and woodstock6.hip4inter.net

<i>Test scenario</i>	<i>Results</i>
Base Exchange (IPv4)	Successful
Base Exchange (IPv6)	Successful
IPv4 readdress OK from wired to wireless	Successful
CLOSE/CLOSE ACK	Successful
Accessing Web service at woodstock4.hip4inter.net through HIP and IPv4	Failure
Accessing Web service at woodstock6.hip4inter.net through HIP and IPv6	Failure

General comments of the HIP implementations

As of May 5, 2008, the newest patch of InfraHIP and hip4inter.net release 2008.04.25 use IP protocol number 139 while the newest OpenHIP version 0.5 still uses IP protocol number 253 for HIP control packets.

Using IPv6 AAAA registers to store the HIT may cause software using IPv6 to automatically try to connect to the IPv6 address, which is actually a HIT.

The Socket *connect()* function may time-out when HIP connections are being established. The OpenHIP communications test in Appendix A describes such a scenario.

OpenHIP criticism

- The HIP kernel patch for Linux is for a several years old kernel version.

- Getting OpenHIP to work in MacOS X is not a trivial task. It requires installing a third-party TAP driver (Boeing Company, 2007). I failed to get it working even though I installed the TAP driver and HIP and verified that they both were running.
- It should be more clearly stated that the IPsec service must be turned off in Windows to be able to use OpenHIP, and that IPv6 must be installed to be able to generate keys. Otherwise the software fails. The installation documentation should inform about this.
- The configuration file for known host often gets truncated for unknown reasons. This can lead to frustration if many IP/HIP mappings have been stored there.

OpenHIP praise

- OpenHIP is currently the only HIP implementation for Mac OS X and Windows.
- This implementation is arguably the easiest to use, because it supports legacy applications out-of-the-box and graphical configuration tools.
- OpenHIP has good documentation.

hip4inter.net criticism

- This software only supports FreeBSD 6.1.
- It does not mention support for retrieving HIP RR from DNS.
- Version 2007.06.11 has a tendency to crash after HIP connections have been terminated.
- I was not able to install newer versions of hip4inter.net on top of 2007.06.11, as the installation failed by a compiler error. This speaks for using this implementation through virtualization software.

hip4inter.net praise

- The included HIP enabled *ping6* tool offers easy testing of HIP Base Exchange and HIP traffic.
- It includes a Java API for HIP.

InfraHip criticism

- Installation is not trivial. I had to install it to directories relative to root (/) instead of the implied location */usr/local* by using the configuration parameter *--prefix=/*, and then manually create symbolic links to the libraries according to error messages reported when running software compiled to use InfraHIP.

- I had problems compiling the test programs in InfraHIP 1.03 that would have been needed to evaluate the software package.

InfraHip praise

- This distribution includes support for HIP network programming through the socket address family AF_HIP.
- InfraHIP has good documentation.
- It includes a Java API for HIP.

DNS extension criticism

The DNS extensions by Boeing could not be used when a Rendezvous Server was specified in the RR description. It caused a configuration parsing error when the HIP service was started.

I find it rather surprising that the only HIP DNS extension in existence (to my knowledge) fails to accept Rendezvous Servers.

I created a workaround patch that enables the use of Rendezvous Servers in the configuration file, and it is available in Appendix A. The patched DNS Bind9 server has been tested and now successfully returns HIP data upon query. An example output is available in Table 15. This patched server provides information about the computers in the Arcada HIP network.

Table 15: Results of a HIP DNS query on one host in the Arcada HIP system

```
bash-2.05b$ dig -t HIP xray.hip.arcada.fi

; <<>> DiG 9.4.1-P1 <<>> -t HIP xray.hip.arcada.fi
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42063
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;xray.hip.arcada.fi.          IN      HIP

;; ANSWER SECTION:
xray.hip.arcada.fi.         900     IN      HIP      2 2001001BA9B870BD79E76FF8FC59FCB6
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC9gx0IFXvhGFhYmoNj
WnRV7R9MB3HKTGKt2HYqnDPr3je7zDKivJWixmEJUC9oi2NXa6nxC5tQ
jLpwkB7jebeHec2nDISiqZ7/11a/orXH71R14oz7fTDBLZNoZ0aCVT7f
vQmI7oBu+rNcyQIzQxbTlGVTDJnWoVcegygpHg3vMwIDAQAB  novy-most.hip.arcada.fi.

;; AUTHORITY SECTION:
hip.arcada.fi.             900     IN      NS       ns.hip.arcada.fi.

;; Query time: 30 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Tue May 6 10:40:03 2008
;; MSG SIZE rcvd: 286
```

LSI addresses

There seems to be different ideas of how to use the LSI address between implementations. OpenHIP treats it as an address that can be used directly to initiate communication between hosts e.g. as the address parameter in a Web browser. HIP4inter.net automatically generates the LSI internally after an application has initiated legacy HIP communication, but does not allow initiating a connection using LSI. This was verified by connection tests using OpenHIP and hip4inter.net.

REFERENCES

- Ahrenholz. (2007, December 6). [*Hipsec*] *HIP implementation tests at IETF 70*. Retrieved May 6, 2008, from <http://www.ietf.org/mail-archive/web/hipsec/current/msg02290.html>
- Ahrenholz, J. (2008, January 11). *HIP DHT Interface, IETF draft*. Retrieved May 6, 2008, from <http://tools.ietf.org/html/draft-ahrenholz-hiprg-dht-02>
- Apple Inc. (2008). *Mac Pro - Tekniset Tiedot*. Retrieved May 5, 2008, from <http://www.apple.com/fi/macpro/specs.html>
- Blanchet, M. (2008, April). *Special-Use IPv6 Addresses, IETF RFC 5156*. Retrieved April 30, 2008, from <http://tools.ietf.org/html/rfc5156#section-2.10>
- Boeing Company. (2007, December 6). Retrieved May 6, 2008, from Boeing HIP server: <http://hipserver.mct.phantomworks.org/>
- Boeing Company. (2007, October 17). *Building from source on OS X in User-mode*. Retrieved May 6, 2008, from http://www.openhip.org/wiki/index.php?title=Building_from_source_on_OS_X_in_User-mode
- Boeing Company. (2007, October 22). *Configuration*. Retrieved April 30, 2008, from OpenHIP: <http://www.openhip.org/wiki/index.php?title=Configuration>
- Boeing Company. (2007, October 22). *Overview*. Retrieved April 30, 2008, from OpenHIP: http://www.openhip.org/wiki/index.php?title=Overview#Linux_kernel_support
- Boeing Company. (2007, October 22). *Usage*. Retrieved May 5, 2008, from OpenHIP: <http://www.openhip.org/wiki/index.php?title=Usage>
- Boeing Company. (2007, October 17). *Windows XP Installation*. Retrieved May 6, 2008, from OpenHIP: http://www.openhip.org/wiki/index.php?title=Windows_XP_Installation

- FreeBSD. (2008, May 5). *FreeBSD as a Guest OS*. Retrieved May 5, 2008, from http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/virtualization-guest.html
- Henderson, T., Nikander, P., & Komu, M. (2007, November 18). *Using the Host Identity Protocol with Legacy Applications, IETF draft*. Retrieved April 30, 2008, from <http://tools.ietf.org/html/draft-ietf-hip-applications-02>
- HIIT. (2008). *Chapter 4. Compiling the Kernel*. Retrieved May 5, 2008, from InfraHIP: <http://infrahip.hiit.fi/hipl/manual/ch04.html>
- HIIT. (2008). *Chapter 5. Compiling Userspace Applications*. Retrieved May 5, 2008, from InfraHIP: <http://infrahip.hiit.fi/hipl/manual/ch05.html>
- HIIT. (2008). *InfraHIP Project: Home*. Retrieved May 5, 2008, from InfraHIP: <http://infrahip.hiit.fi/>
- HIIT. (2008). *InfraHIP Project: Links*. Retrieved May 6, 2008, from <http://www.infrahip.net/index.php?index=links>
- Internet Assigned Numbers Authority. (2008, April 18). *Assigned Internet Protocol Numbers*. Retrieved April 30, 2008, from <http://www.iana.org/assignments/protocol-numbers>
- Jokela, P., Moskowitz, R., & Nikander, P. (2008, April). *Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP), IETF RFC 5202*. Retrieved April 30, 2008, from <http://tools.ietf.org/html/rfc5202>
- Josefsson, S. (2003, July). *RFC 3548 - The Base16, Base32, and Base64 Data Encodings*. (S. Josefsson, Ed.) Retrieved April 30, 2008, from <http://www.faqs.org/rfcs/rfc3548.html>
- Kent, S. (2005, December). *IP Encapsulating Security Payload (ESP), IETF RFC 4303*. Retrieved April 30, 2008, from <http://tools.ietf.org/html/rfc4303>
- Komu, M., & Henderson, T. (2008, February 25). *Basic Socket Interface Extensions for Host Identity Protocol (HIP), IETF draft*. Retrieved April 30, 2008, from <http://tools.ietf.org/html/draft-ietf-hip-native-api-04>

- Komu, M., Henderson, T., Matthews, P., Tschofenig, H., Keraenen, A., Melen, J., et al. (2008, February 25). *Basic HIP Extensions for Traversal of Network Address Translators and Firewalls, IETF draft*. Retrieved April 30, 2008, from <http://tools.ietf.org/html/draft-ietf-hip-nat-traversal-03>
- Krasnyansky, M. (2000). *Universal TUN/TAP device driver*. Retrieved April 30, 2008, from <http://www.kernel.org/pub/linux/kernel/people/marcelo/linux-2.4/Documentation/networking/tuntap.txt>
- Laganier, J., & Eggert, L. (2008, April). *Host Identity Protocol (HIP) Rendezvous Extension, IETF RFC 5204*. Retrieved April 30, 2008, from <http://tools.ietf.org/html/rfc5204>
- Laganier, J., Koponen, T., & Eggert, L. (2008, April). *Host Identity Protocol (HIP) Registration Extension, IETF RFC 5203*. Retrieved April 30, 2008, from <http://tools.ietf.org/html/rfc5203>
- McGregor, A. (2007, July 19). *Installation Instructions for PyHIP*. Retrieved May 6, 2008, from <http://www.sharemation.com/adm01bass/pyhip-2003-07-19.tar.bz2/README>
- Moskowitz, R., & Nikander, P. (2006, May). *Host Identity Protocol (HIP) Architecture, IETF RFC 4423*. Retrieved April 30, 2008, from <http://tools.ietf.org/html/rfc4423>
- Moskowitz, R., Nikander, P., Jokela, P., & Henderson, T. (2008, April). *Host Identity Protocol, IETF RFC 5201*. Retrieved April 30, 2008, from <http://tools.ietf.org/html/rfc5201>
- Nikander, P., & Laganier, J. (2008, April). *Host Identity Protocol (HIP) Domain Name System (DNS) Extension, IETF RFC 5205*. Retrieved April 30, 2008, from <http://tools.ietf.org/html/rfc5205>
- Nikander, P., Henderson, T., Vogt, C., & Arkko, J. (2008, April). *End-Host Mobility and Multihoming with the Host Identity Protocol, IETF RFC 5206*. Retrieved April 30, 2008, from <http://tools.ietf.org/html/rfc5206>

Nikander, P., Laganier, J., & Dupont, F. (2007, April). *An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers (ORCHID)*, IETF RFC 4843. Retrieved April 30, 2008, from <http://tools.ietf.org/html/rfc4843>

Oy LM Ericsson Ab. (2008, April 30). *HIP for BSD Project: Download*. Retrieved May 6, 2008, from <http://hip4inter.net/download/download.php>

Oy LM Ericsson Ab. (2007, June 11). *HIP for BSD project: FAQ*. Retrieved May 6, 2008, from HIP4Inter.net: <http://hip4inter.net/faq.php>

Oy LM Ericsson Ab. (2005, November 2). *HIP Implementation for FreeBSD*. Retrieved April 30, 2008, from <http://www.hip4inter.net/documentation/hip4bsd.pdf>

Parallels. (2008, May 5). *Parallels Desktop 3.0 for Mac*. Retrieved May 5, 2008, from Parallels: http://www.vmware.com/products/fusion/system_requirements.html

Parallels. (2008). *System Requirements*. Retrieved May 5, 2008, from Parallels Desktop for Mac: http://www.parallels.com/en/products/desktop/sr/#guest_os

SourceForge. (2008, May 6). *OpenHIP patches*. Retrieved May 6, 2008, from SourceForge: http://sourceforge.net/project/showfiles.php?group_id=132288&package_id=152401

VMware. (2007). *Fusion Datasheet*. Retrieved May 5, 2008, from http://www.vmware.com/files/pdf/fusion_datasheet.pdf

VMware. (2008, May). *Fusion System Requirements*. Retrieved May 5, 2008, from http://www.vmware.com/products/fusion/system_requirements.html

APPENDIX A: A HIP COMMUNICATION EXAMPLE

The firewall can present a problem, opening protocols 50 (ESP), 253 and 139 (HIP) should allow HIP traffic.

The performance of a computer may play a role. When a connection between two OpenHIP computers was tested, and one computer was a 500 MHz laptop, the connection timed out before the HIP handshake was completed. After changing to a 1.6 GHz laptop the connection was successfully established without timeout.

OpenHIP testing

Test of a HIP connection between a laptop and a desktop computer. Both use Windows XP.

To be able to connect to each other, the remote computer's HIP, IP and LSI are inserted into the *known_host_identities.xml* file. To contact the other computer, the LSI is then used in the same manner as an ordinary IPv4 address.

In this test, a simple peer-to-peer chat program was used to get plain text messages that can be easily interpreted. Figure A1 shows an encrypted ESP packet, no plain text message can be seen in the hex dump although the message sent was a plain text message.

The test was first run with default configuration, with encryption enabled in ESP.

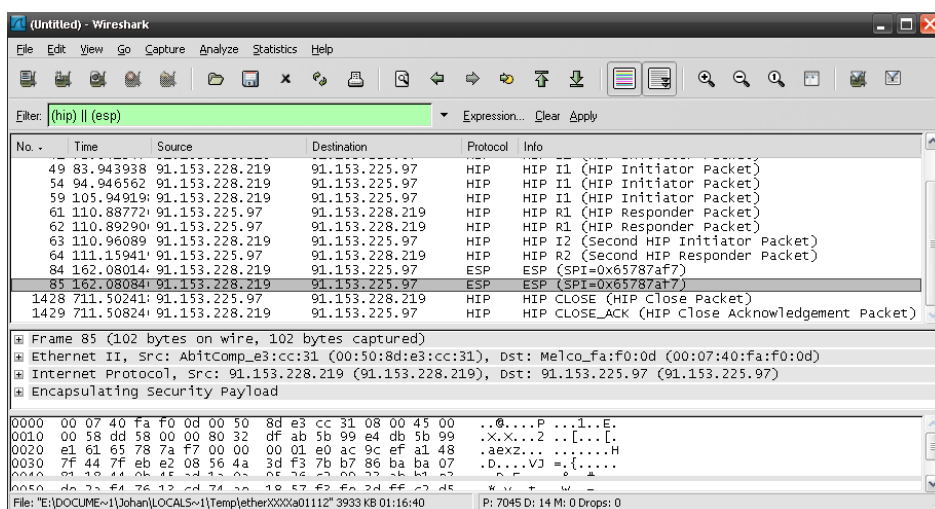


Figure A1: Communication between two Windows XP computers running OpenHIP 0.5 with encryption is captured using Wireshark

In the second test plain-text communicating was tested by preferring encryption suites 5 and 6 which are NULL encryption suites (Table A1). Changing locator was also tested.

Table A1: Segment from the OpenHIP configuration file hip.conf where NULL encryption is preferred over encryption

```
<esp_sa>
  <transforms>
    <id>5</id>
    <id>6</id>
    <id>1</id>
    <id>2</id>
    <id>3</id>
    <id>4</id>
  </transforms>
</esp_sa>
```

The chat software is tinyp2p by Mike Tsao, available from <http://www.sowbug.org/mt/2003/06/tinyp2p-a-ridiculously-simple.html> (retrieved April 12, 2008). It has a simple user interface (Figure A2).

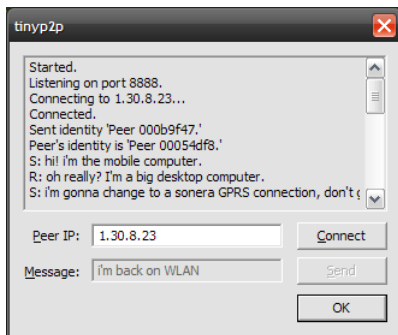


Figure A2: The user interface in the chat program tinyp2p

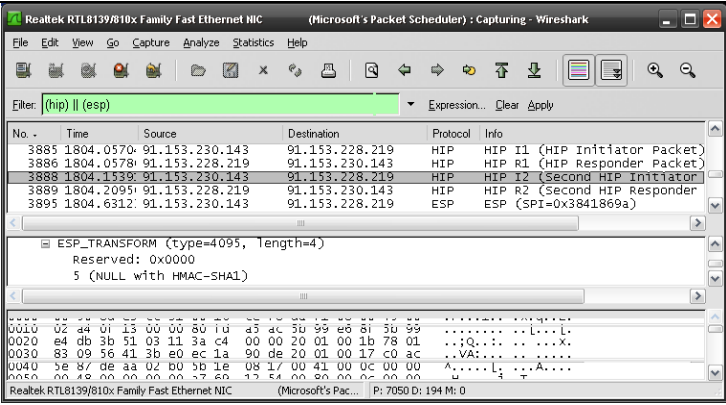
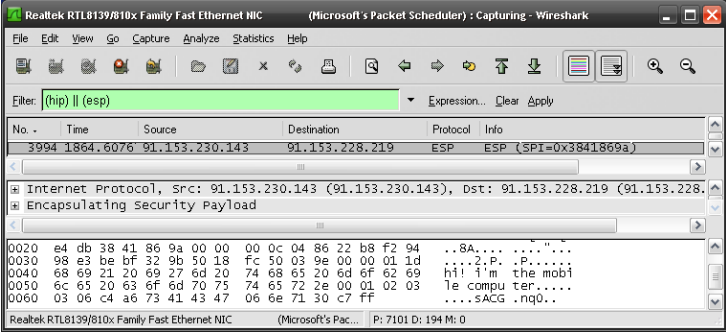
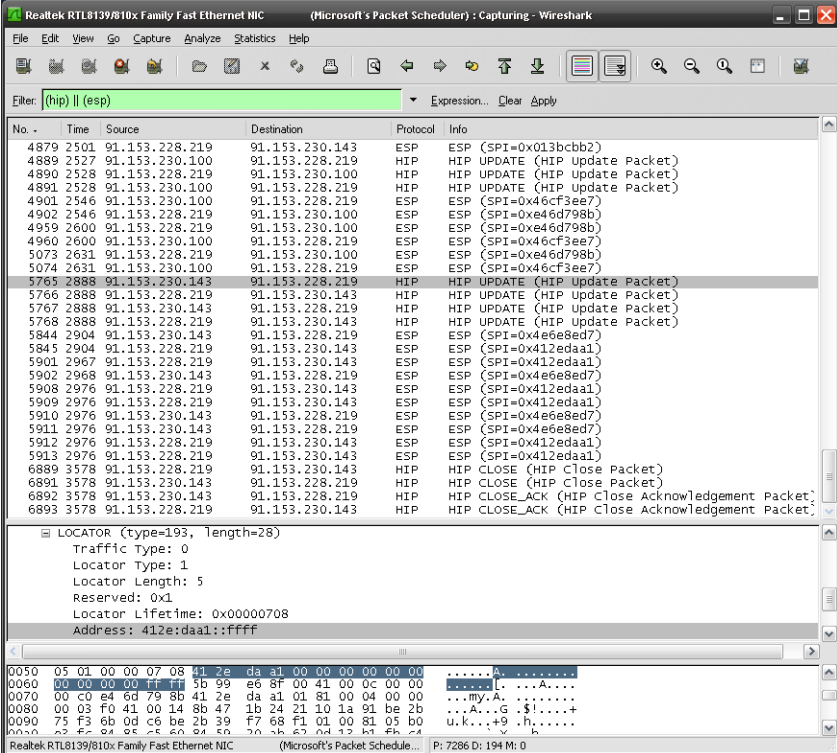
The dialog between the computers was recorded and is presented in Table A2. It illustrates how the physical connection of nodes in HIP communication can be temporarily disables or changed to another IP address, e.g. when moving between LAN and WLAN.

The same sequence was monitored through the packet capture program Wireshark, and shows what goes on at the protocol level (Table A3). Because we used NULL encryption we are able to see the messages in the capture window.

Table A2: Dialog between two computers chatting over a HIP connection while the mobile computer changes locator

<i>Description</i>	<i>Mobile computer chat window</i>	<i>Desktop computer chat window</i>
The mobile computer contacts the desktop computer's LSI and types a message. The accept address seen in the desktop computer's chat window is the mobile computers LSI.	Started. Listening on port 8888. Connecting to 1.30.8.23... Connected. Sent identity 'Peer 000b9f47.' Peer's identity is 'Peer 00054df8.' S: hi! i'm the mobile computer.	Accepted connection from 1.26.144.222. Sent identity 'Peer 00054df8.' Peer's identity is 'Peer 000b9f47.' R: hi! i'm the mobile computer.
The desktop computer answers.	R: oh really? I'm a big desktop computer.	S: oh really? I'm a big desktop computer.
The mobile computer informs about the intention to switch to GRPS.	S: i'm gonna change to a sonera GPRS connection, don't go anywhere	R: i'm gonna change to a sonera GPRS connection, don't go anywhere
The mobile computer has switched to a GRPS internet connection, and sends out messages. The messages are not getting through because some node in the Sonera GRPS network is not forwarding HIP packets with IP protocol 253.	S: ok, i'm on GPRS now. S: hello? S: (i'm gonna change back to WLAN. he's not getting me)	
The mobile computer changes back to WLAN.	R: did you say anything? S: never mind.	S: did you say anything? R: never mind.
The mobile computer changes to LAN from WLAN successfully, and the connection is not disturbed.	S: i'm testing ordinary LAN now, i'll disconenct the WLAN. S: ok now i'm on LAN.	R: i'm testing ordinary LAN now, i'll disconenct the WLAN. R: ok now i'm on LAN.
	R: i didn't notice a thing. nice! very nice. R: ok this is wrapped up. goodbye.	S: i didn't notice a thing. nice! very nice. S: ok this is wrapped up. goodbye.
The mobile computer successfully switches back to WLAN. The desktop computer closes the chat program.	S: i'm back on WLAN R: I'm quitting. Bye. Peer is disconnecting. Goodbye! Peer closed connection.	R: i'm back on WLAN S: I'm quitting. Bye.

Table A3: Wireshark capture of a chat session between two computers, using the HIP protocol

Description	Screenshot
<p>The connection is established when the mobile computer uses the connect function in the chat program. Notice that we are using NULL encryption, because we preferred it in the OpenHIP configuration file.</p>	
<p>The message sent from the mobile computer can be seen in plain text in the ESP packet.</p>	
<p>In this screenshot we can see the sequence of packets sent when the mobile computer switches from WLAN to LAN and then back to WLAN. This can be verified by observing the addresses in the source and destination columns, they change after the UPDATE packets. Notice the LOCATOR parameter in the first UPDATE packet. It contains the updated IP address in a 128 bit field. The 32 bit IP address is followed by zeros.</p>	

APPENDIX B: PATCH TO BYPASS CONFIGURATION PARSING ERRORS IN THE BIND SERVER WHEN RENDEZVOUS SERVERS ARE DESCRIBED

This patch is for bind-9.4.1, but may be applied to earlier versions as well.

Edit the source file bind-9.4.1/lib/isc/base64.c, function `isc_base64_tobuffer`(line179) .

- define *int* variable *badcode*
- after the line `tr = &token.value.as_textregion;` add the following lines:

```
badcode=0;
for (i = 0; i < tr->length; i++)
    if( !((tr->base[i] >= 'A' && tr->base[i] <= 'Z') || (tr->base[i] >= 'a' && tr->base[i]
<= 'z') || (tr->base[i] >= '0' && tr->base[i] <= '9') || tr->base[i] == '/' || tr-
>base[i] == '+') ) badcode=1;
if(badcode==1) break;
```

These modifications will make it possible to add Rendezvous Servers to the HIP RR without the Bind server reporting parse errors and terminating on start-up. The problem is that the Bind parser will fail to detect when the Base64 code of the Host Identity has ended. This patch adds a check for this condition.

APPENDIX C: OVERVIEW OF THE CONTENTS OF THE HIP SPECIFICATIONS

The Internet Engineering Task Force IETF hosts 10 documents describing the experimental Host Identity Protocol (HIP), as well as references related documents. HIP is not standardized as of May 5, 2008. 7 of the 10 are released as numbered RFC documents while the remaining three are drafts that are actively updated by the HIP working group. As of May 5, 2008, all the documents are tagged as experimental except for RFC4423.

An index of all HIP documents and links to them is available on <http://www.ietf.org/html.charters/hip-charter.html>. All WWW links below were retrieved on April 30, 2008.

- RFC4423: Host Identity Protocol (HIP) Architecture (<http://tools.ietf.org/html/rfc4423>)
- RFC5201: Host Identity Protocol (<http://tools.ietf.org/html/rfc5201>)
- RFC5205: Host Identity Protocol (HIP) Domain Name System (DNS) Extension (<http://tools.ietf.org/html/rfc5205>)
- RFC5202: Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP) (<http://tools.ietf.org/html/rfc5202>)
- RFC5206: End-Host Mobility and Multi-homing with the Host Identity Protocol (<http://tools.ietf.org/html/rfc5206>)
- RFC5203: Host Identity Protocol (HIP) Registration Extension (<http://tools.ietf.org/html/rfc5203>)
- RFC5204: Host Identity Protocol (HIP) Rendezvous Extension (<http://tools.ietf.org/html/rfc5204>)
- Basic HIP Extensions for Traversal of Network Address Translators and Firewalls (<http://tools.ietf.org/html/draft-ietf-hip-nat-traversal-03>)
- Basic Socket Interface Extensions for Host Identity Protocol (HIP) (<http://tools.ietf.org/html/draft-ietf-hip-native-api-04>)
- Using the Host Identity Protocol with Legacy Applications (<http://tools.ietf.org/wg/hip/draft-ietf-hip-applications>)

RFC4423: Host Identity Protocol (HIP) Architecture

This document is more conceptual than the other, describing the HIP architecture in general. It describes the key concepts of the protocol; the methods by which host identities are used and how HIP as a concept would work within the constraints of the real Internet. Solutions to problems that these constraints produce, such as the Local Scope Identifier in legacy applications and how HIP deals with the problem of being used from a network using NAT (Network Address Translation), are discussed. This document also briefly describes all the other topics that are discussed in more detail in the technical specifications for the different components.

RFC5201: Host Identity Protocol

This document describes in detail about the host identifiers and tags and generating them, the mechanisms used in the Base Exchange, and generally topics related to establishing and maintaining a connection.

RFC5205: Host Identity Protocol (HIP) Domain Name System (DNS) Extension

This document describes and specifies how a Host Identity, a Host Identity Tag, and rendezvous information are stored in DNS records.

It is worth noticing that as of April 2008 few implementations of HIP actually retrieve HIP information as described in this document. A more common solution is to use a local file that maps IP addresses and Host Identity Tags, or to store the HIT as an IPv6 (AAAA) record in DNS.

Distributed Hash Tables (DHT, see <http://www.opendht.org/>) are used as an alternative to DNS in many HIP implementations. InfraHIP, OpenHIP and hip4inter.net all have some degree of support for DHT, or at least non-functional references to DHT in the source code or included tools. In DHT a HIT is used as a search key to look up an IP address or vice versa. DHT is not part of the IETF specifications of HIP.

RFC5202: Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP)

This document describes and specifies the use of ESP (Encapsulating Security Payload) to transmit data traffic between hosts using HIP, after they have established security associations through the Base Exchange.

RFC5206: End-Host Mobility and Multihoming with the Host Identity Protocol

The End-Host Mobility part describes how a host communicating with another host can update its locator (i.e. IP address), and then inform the other host of its new address, and in particular how ESP/IPsec security associations are managed after the IP address changes.

Multihoming discusses how a host can have several IP addresses for one Host Identity, and concepts related to this such as preferred locators.

RFC5203: Host Identity Protocol (HIP) Registration Extension

This document describes and specifies how a client using HIP can request registration for a service e.g. to use a rendezvous service or HIP relay. The registration is done by sending HIP control packets to the server hosting the service.

RFC5204: Host Identity Protocol (HIP) Rendezvous Extension

This document describes and specifies techniques to use a server that maintains a database over IP/HIT pairs. Usage of a Rendezvous Server requires registration as described in RFC5203. When a host queries the DNS service for a HIP RR and the RR contains a Rendezvous Server entry, the HIP initiation packet should be sent to the Rendezvous Server instead of to the host. The Rendezvous Server further forwards the packet to the destination host. Hosts registered to a Rendezvous Server should keep the Rendezvous Server up-to-date with valid IP addresses.

Basic HIP Extensions for Traversal of Network Address Translators and Firewalls

This document describes how Rendezvous Servers are used for a secondary purpose in relaying UDP packed HIP control packets (TCP packed data is not yet supported). This is a proposed solution to the problem of accessing a HIP node that is inside a local NAT network or behind a firewall. A typical problematic situation is when a NAT device cannot resolve incoming connections to the intended recipient. This document also describes how the hosts can use ICE techniques when trying to establish direct connection between each other. ICE is a technique developed by Cisco for analyzing possible direct routes between hosts (Rosenberg, 2007). A version of ICE for any UDP application is also available (Rosenberg, 2008a), as well as propositions for a version of ICE compatible with TCP (Rosenberg, 2008b).

Basic Socket Interface Extensions for Host Identity Protocol (HIP)

This document describes HIP extensions to the Socket API, i.e. the function calls and structures used by the HIP Socket extensions. It also describes techniques used by hosts to retrieve the HIT of a host.

Using the Host Identity Protocol with Legacy Applications

This document describes how applications that do not have native HIP capabilities could be used through HIP. (Henderson, Nikander, & Komu, 2007)

References

Rosenberg, J. (2007, October 29). *Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols, IETF Draft*. Retrieved April 30, 2008, from <http://tools.ietf.org/html/draft-ietf-mmusic-ice-19>

Rosenberg, J. (2008a, February 15). *NICE: Non Session Initiation Protocol (SIP) usage of Interactive Connectivity Establishment (ICE), IETF draft*. Retrieved April 30, 2008, from <http://tools.ietf.org/html/draft-rosenberg-mmusic-ice-nonsip-00>

Rosenberg, J. (2008b, February 25). *TCP Candidates with Interactive Connectivity Establishment (ICE), IETF draft*. Retrieved April 30, 2008, from <http://tools.ietf.org/html/draft-ietf-mmusic-ice-tcp-06>

APPENDIX D: IDEAS FOR A WINDOWS HIP SERVICE

A Windows native HIP driver seems to be plausible using kernel/user mode drivers that operate in mixed mode. I initially proposed to develop a product using the methods presented here for a company that considered them plausible. I have not tested the methods in practice but such a test could provide some ideas for a well performing Windows HIP driver.

By exploiting existing Windows components it could be possible to develop a Windows HIP driver that not only performs well, but does not require licenses of commercial cryptographic functions and is not restrained to Open Source licenses such as those enforced by the OpenSSL package.

Windows XP and Vista have built-in cryptography functions, including IPsec and ESP, which could possibly be exploited to do the ESP packet handling and encryption as stated in the HIP specifications. It would require further research to verify if this is indeed possible.

The Windows operating system defines so-called intermediate device drivers that are injected between layers of the operating system network stack. By operating directly on IP packets it could be possible to modify the packets as needed with the use of mixed-mode device drivers. When an incoming IP packet is received it is analyzed. If the IP address is already mapped in an existing HIP communication it is passed to the HIP user mode application for further processing, after which the IP packet is discarded. When a packet is sent through the native HIP API it is directly injected through the HIP device driver into the outgoing IP packet stack. Legacy support is done by detecting outgoing packets that match the LSI akin to an IPv4 address or HIT akin to an IPv6 address. LSI support would only match already established connections because supporting the initiation of HIP communication using LSI is not stated in the specifications and is quite pointless in my opinion. Establishing communication through HIT would first perform a DNS lookup to see if the HIT is valid and to get the IP address or Rendezvous Server from where to get the IP address.

This Windows version would not support any unspecified gimmicks that are used in other implementations solely for the purpose of experimentation, which could make the use of the software un-trivial.

The driver should only use the network drivers to perform data injection and detecting HIP packets. All processing would be done in the user mode HIP service.

The information about established communications could be stored in efficient data structures in the memory of the HIP service.

APPENDIX E: INTERNET ADDRESSES FOR HIP RELATED SOFTWARE AND INFORMATION

Table E1 lists some WWW links so web-sites related to HIP. Table E2 list download links for the software discussed in this report. The links were valid May 6, 2008.

Table E1: Internet links to project information and documentation

<i>Description</i>	<i>URL</i>
Arcada HIP domain entry point	http://hip.arcada.fi/
Hip4inter.net project homepage, online documentation and download links	http://hip4inter.net/
Hip4inter.net documentation	http://www.hip4inter.net/documentation/hip4bsd.pdf
InfraHIP project homepage, online documentation and download links	http://infrachip.hiit.fi/
InfraHIP online documentation	http://infrachip.hiit.fi/hipl/manual/index.html
OpenHIP project homepage, online documentation and download links	http://www.openhip.org/
OpenHIP online documentation	http://www.openhip.org/wiki/index.php?title=Main_Page
Boeing HIP server	http://hipserver.mct.phantomworks.org/
HIP specifications	http://tools.ietf.org/wg/hip/

Table E2: Download links to HIP related software

InfraHIP 1.03 distribution package	http://infrachip.hiit.fi/index.php?index=download
OpenHIP 0.5 distribution package	http://sourceforge.net/project/showfiles.php?group_id=132288&package_id=223220
Hip4inter.net	http://hip4inter.net/ (requires license agreement)
PyHIP distribution package	http://www.sharemation.com/adm01bass/pyhip/
Patch to add HIP support to Wireshark	http://sourceforge.net/project/showfiles.php?group_id=132288&package_id=152401&release_id=501589
Patch to add HIP RR support to the Bind DNS server	http://openhip.cvs.sourceforge.net/openhip/patches/bind/

APPENDIX F: LIST OF HIP TEST SERVERS

It is implied that these servers run WWW services, but this has not been verified. The sources for this information are <http://hip4inter.net/faq.php>, <http://hipserver.mct.phantomworks.org> and <http://www.infracorp.net/index.php?index=links>.

The typically work without a HIP enabled connection, presenting a notice about HIP not being in use.

Table F1 shows test servers that host WWW pages through HIP. The links and references were valid May 6, 2008.

Table F1: HIP test servers

<i>Description</i>	<i>URL</i>
HIP test server for hip4inter.net	http://woodstock4.hip4inter.net
HIP test server for hip4inter.net, IPv6	http://woodstock6.hip4inter.net
Infra HIP test server	http://hipserver.infracorp.net
Infra HIP test server	http://crossroads.infracorp.net
Boeing HIP test server	http://hipserver.mct.phantomworks.org/

DEGREE THESIS

JOHAN FRÖJDMAN