Internet Research Task Force (IRTF)
Request for Comments: 6538
Category: Informational
ISSN: 2070-1721

T. Henderson
The Boeing Company
A. Gurtov
University of Oulu

March 2012

The Host Identity Protocol (HIP) Experiment Report

Abstract

This document is a report from the IRTF Host Identity Protocol (HIP) research group documenting the collective experiences and lessons learned from studies, related experimentation, and designs completed by the research group. The document summarizes implications of adding HIP to host protocol stacks, Internet infrastructure, and applications. The perspective of a network operator, as well as a list of HIP experiments, are presented as well. Portions of this report may be relevant also to other network overlay-based architectures or to attempts to deploy alternative networking architectures.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Research Task Force (IRTF). The IRTF publishes the results of Internet-related research and development activities. These results might not be suitable for deployment. This RFC represents the consensus of the IRTF HIP Research Group of the Internet Research Task Force (IRTF). Documents approved for publication by the IRSG are not a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6538.

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction

This document summarizes the work and experiences of the IRTF's Host Identity Protocol research group (HIP-RG) in the 2004-2009 time frame. The HIP-RG was chartered to explore the possible benefits and consequences of deploying the Host Identity Protocol architecture [RFC4423] in the Internet and to explore extensions to HIP.

This document was developed over several years as the main charter item for the HIP research group, and it has received inputs and reviews from most of the active research group participants. There is research group consensus to publish it.

1.1. What is HIP?

The Host Identity Protocol architecture introduces a new namespace, the "host identity" namespace, to the Internet architecture. The express purpose of this new namespace is to allow for the decoupling of identifiers (host identities) and locators (IP addresses) at the internetworking layer of the architecture. The contributors to HIP have expected that HIP will enable alternative solutions for several of the Internet's challenging technical problems, including potentially host mobility, host multihoming, site multihoming, IPv6 transition, NAT traversal, and network-level security. Although there have been many architectural proposals to decouple identifiers and locators over the past 20 years, HIP is one of the most actively developed proposals in this area [book.gurtov].

The Host Identity Protocol itself provides a rapid exchange of host identities (public keys) between hosts and uses a Diffie-Hellman key exchange that is compliant with Sigma ("SIGn-and-MAc") to establish shared secrets between such endpoints [RFC5201]. The protocol is designed to be resistant to Denial-of-Service (DoS) and Man-in-the-Middle (MitM) attacks, and when used together with another suitable security protocol, such as Encapsulated Security Payload (ESP) [RFC4303], it provides encryption and/or authentication protection for upper-layer protocols such as TCP and UDP, while enabling continuity of communications across network-layer address changes.

A number of Experimental RFC specifications were published by the IETF's HIP working group, including the HIP base protocol [RFC5201], ESP encapsulation [RFC5202], registration extensions [RFC5203], HIP rendezvous servers [RFC5204], DNS resource records [RFC5205], and mobility management [RFC5206]. Host identities are represented as Overlay Routable Cryptographic Hash Identifiers (ORCHIDs) [RFC4843] in Internet protocols. Additionally, the research group published one RFC on requirements for traversing NATs and firewalls [RFC5207]

and the working group later published specification text for legacy NAT traversal [RFC5770]. As of this writing, work has commenced on moving the above specifications to Standards Track status.

1.2. Terminology

The terms used in this document are defined elsewhere in various documents. In particular, readers are suggested to review Section 3 of [RFC4423] for a listing of HIP-specific terminology.

1.3. Scope

The research group has been tasked with producing an "experiment report" documenting the collective experiences and lessons learned from other studies, related experimentation, and designs completed by the research group. The question of whether the basic identifier-locator split assumption is valid falls beyond the scope of this research group. When indicated by its studies, the HIP-RG can suggest extensions and modifications to the protocol and architecture. It has also been in scope for the RG to study, in a wider sense, what the consequences and effects that wide-scale adoption of any type of separation of the identifier and locator roles of IP addresses is likely to have.

During the period of time when the bulk of this report was drafted (2004-2009), several research projects and open source software projects were formed to study HIP. These projects have been developing software enabling HIP to be interoperable according to the Experimental RFCs as well as supporting extensions not yet specified by RFCs.

The research group has been most active in two areas. First and foremost, the research group has studied extensions to HIP that went beyond the scope and charter of the IETF HIP working group and the set of RFCs (RFC 5201-5206) initially published in April 2008. Some of this work (NAT traversal, certificate formats for HIP, legacy application support, and a native sockets API for HIP) ultimately flowed into the IETF HIP working group upon its recharter in 2008. Other extensions (e.g., HIP in the Internet Indirection Infrastructure (i3) overlay, use of distributed hash tables for HIT-based (Host Identity Tag) lookups, mobile router extensions, etc.) are either still being worked on in the research group or have been abandoned. Most of the energy of the research group during this time period has been in studying extensions of HIPs or the application of HIP to new problem domains (such as the Internet of Things). Second, the research group has discussed the progress and outcome of the implementations and experiments conducted so far, as well as discussing perspectives from different participants (end users,

operators, enterprises) on HIP deployment. It is this latter category of work (and not the extensions to HIP) on which this report is focused.

Finally, the research group was chartered to study, but did not rigorously study (due to lack of inputs), the following issues:

- Objective comparisons of HIP with other mechanisms (although the research group did hold some discussions concerning the relation of HIP to other efforts such as the End-Middle-End (EME) research group, the Routing research group (RRG), and shim6-based protocols).

- Large scale deployments (thousands of hosts or greater).

- Exploration of whether introducing an identity-locator mechanism would be architecturally sound, deployed at wide scale.

- Changes to the HIP baseline architecture and protocol or other identity-locator separation architectures.

1.4. Organization

In this report, we summarize the collective experience of early implementers and adopters of HIP, organized as follows:

- Section 2 describes the implications of supporting HIP on an end host.

- Section 3 covers a number of issues regarding the deployment of and interaction with network infrastructure, including middlebox traversal, name resolution, Access Control Lists (ACLs), and HIP infrastructure such as rendezvous servers.

Whereas the two previous sections focus on the implementation and deployment of the network plumbing to make HIP work, the next three focus on the impact on users and operators of the network.

- Section 4 examines how the support of HIP in the host and network infrastructure affects applications; whether and how HIP provides benefits or drawbacks to HIP-enabled and legacy applications.

- Section 5 provides an operator's perspective on HIP deployment.

- Section 6 discusses user privacy issues.

In closing, in Section 7, we list the experimental activities and research that have contributed to this report, and in Section 8 we briefly summarize related work.

2. Host Stack Implications

HIP is primarily an extension to the TCP/IP stack of Internet hosts, and, in this section, we summarize some experiences that several implementation groups have encountered in developing this extension. Our discussion here draws on experience of implementers in adding HIP to general-purpose computing platforms such as laptops, desktops, servers, and PDAs. There are two primary ways to support HIP on such an end host. The first is to make changes to the kernel implementation to directly support the decoupling of identifier and locator. Although this type of modification has data throughput performance benefits, it is also the more challenging to deploy. The second approach is to implement all HIP processing in the user-space and configure the kernel to route packets through user-space for HIP processing.

The following public HIP implementations are known and actively maintained:

- HIP4BSD (http://www.hip4inter.net) -- FreeBSD kernel modifications and user-space keying daemon;
- HIPL (http://hipl.hiit.fi) -- Linux kernel and user-space implementation;
- OpenHIP (http://www.openhip.org) -- User-space keying daemon and packet processing for Linux, Windows XP/Vista/7, and Apple OS X.

In the following, we first describe issues specific to the way in which HIP is added to a stack, then we discuss general issues surrounding both implementation approaches.

2.1. Modifications to TCP/IP Stack Implementations

In this section, we focus on the support of HIP assuming the following:

- HIP is implemented by directly changing the TCP/IP stack implementation.
- Applications (using the sockets API) are unaware of HIP.

A HIP implementation typically consists of a key management process that coordinates with an IPsec-extended stack, as shown in Figure 1. In practice, HIP has been implemented entirely in the user-space, entirely in the kernel, or as a hybrid with a user-space key management process and a kernel-level ESP.

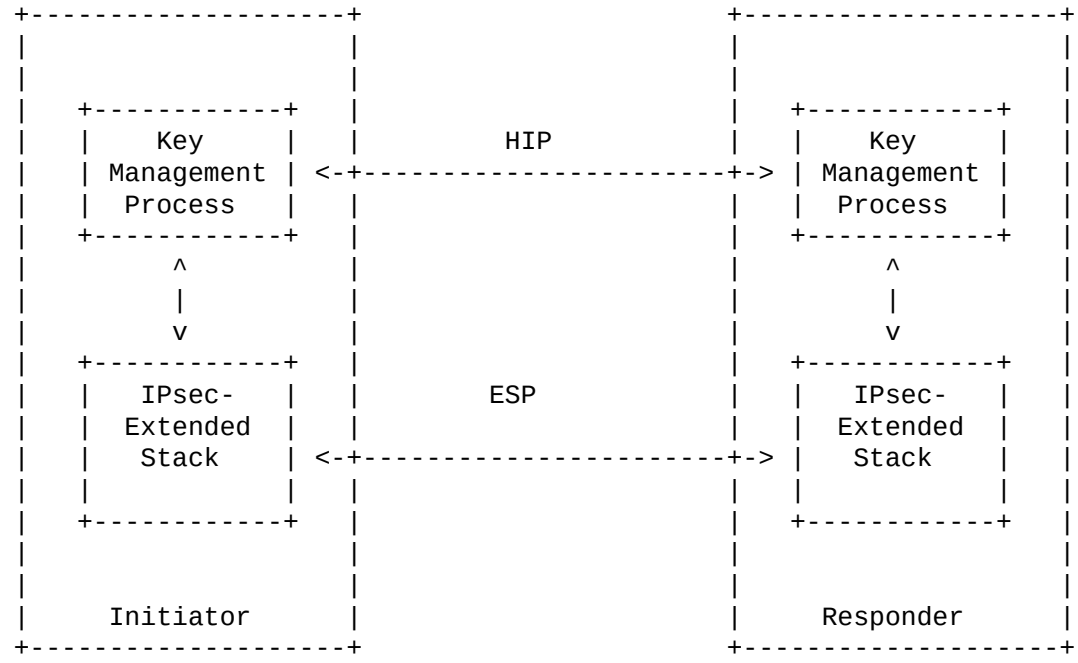

Figure 1: HIP Deployment Model

Figure 2 summarizes the main implementation impact of supporting HIP, and is discussed further in subsequent sections. To enable HIP natively in an implementation requires extensions to the key management interface (here depicted as PF_KEY API [RFC2367]) with the security association database (SAD) and security policy database (SPD). It also requires changes to the ESP implementation itself to support BEET-mode (Bound End-to-End Tunnel) processing [BEET-MODE], extensions to the name resolution library, and (in the future) interactions with transport protocols to respond correctly to mobility and multihoming events [TCP-RLCI].

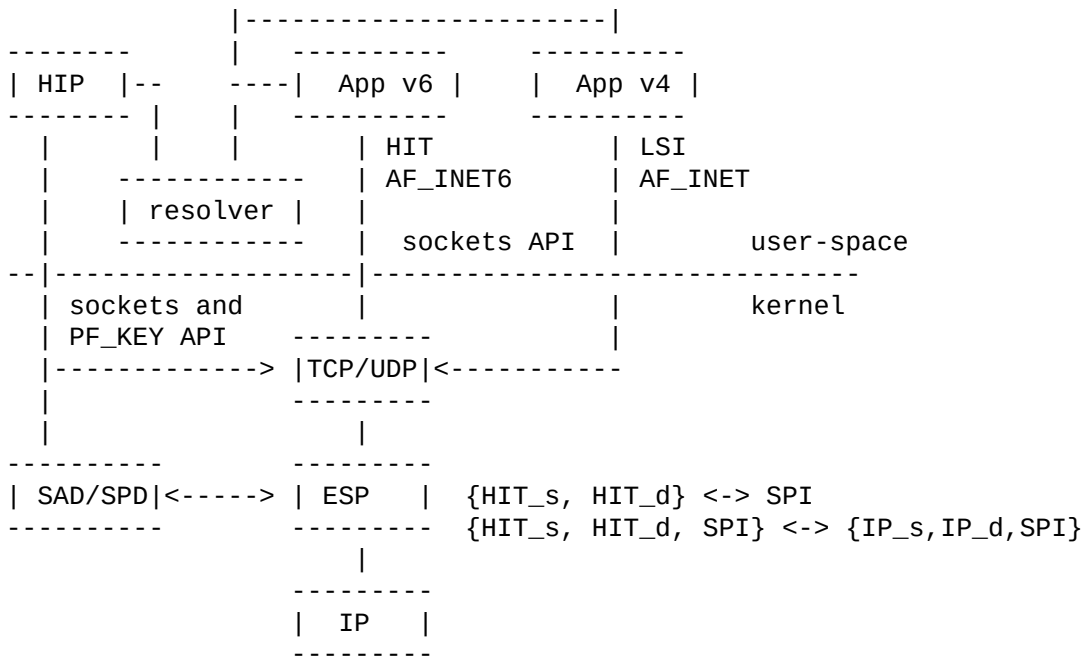

Figure 2: Overview of Typical Implementation Changes to Support HIP

Legacy applications can continue to use the standard AF_INET6 (for IPv6) and AF_INET (for IPv4) sockets API. IPv6 applications bind directly to a Host Identity Tag (HIT), which is a part of IPv6 address space reserved for ORCHIDs. IPv4 applications bind to a Local Scope Identifier (LSI) that has significance only within a host; the HIP layer translates from LSIs and HITs to the IP addresses that are still used underneath for HIP base exchange.

2.1.1. ESP Implementation Extensions

HIP uses a Bound End-to-End Tunnel (BEET) mode of ESP operation, which mixes tunnel-mode semantics with transport-mode syntax. BEET is not supported by all operating system distributions at present, so kernel modifications might be needed to obtain true kernel support using existing IPsec code. At the time of writing, the BEET mode has been adopted to vanilla Linux and FreeBSD kernels.

The HIPL project has contributed an IPsec BEET patch for the Linux kernel. The kernel-level support could potentially allow all Linux implementations of HIP to run in the user-space and use a common interface towards the kernel.

One inconvenience experienced in current Linux IPsec implementation (due to the native IPsec implementation, not HIP specifically) is a loss of the first data packet that triggers the HIP association establishment. Instead, this packet should be cached and transmitted after the association is established.

2.2. User-Space Implementations

HIP can be implemented entirely in the user-space, an approach that is essential for supporting HIP on hosts for which operating system modifications are not possible. Even on modifiable operating systems, there is often a significant deployment advantage in deploying HIP only as a user-space implementation. All three open-source implementations provide user-space implementations and binary packages (RPMs, DEBs, self-extracting installers) typical of application deployment on the target systems.

When HIP is deployed in the user-space, some technique is necessary to identify packets that require HIP processing and divert them to the user-space for such processing and to re-inject them into the stack for further transport protocol processing. A commonly used technique is to deploy a virtual device in the kernel such as a network tap (TAP) device, although operating systems may provide other means for diverting packets to user-space. Routing or packet filtering rules must be applied to divert the right packets to these devices.

As an example, the user-space implementation may install a route that directs all packets with destination addresses corresponding to HITs or LSIs to such a virtual device. In the user-space daemon, the ESP header and possibly the UDP header is applied, an outer IP address replaces the HIT, and the packet is re-sent to the kernel. In the reverse direction, a socket associated to ESP or a UDP port number may be used to receive ESP-protected packets. HIP signaling packets themselves may be sent and received by a raw socket bound to the HIP number or UDP port when UDP encapsulation is used.

2.3. Issues Common to Both Implementation Approaches

2.3.1. User-Space Handling of HITs

Much initial experimentation with HIP has involved using HITs directly in IPv6 socket calls, without any resolution infrastructure to learn the HIT based on, for example, a domain name, or to resolve the IP address. To experiment with HIP using HITs requires a priori HIT exchange, in the absence of a resolution service. Manual exchange of HITs has been a major inconvenience for experimentation. It can be overcome via 1) opportunistic HIP mode (RFC 5201, Section

4.1.6), 2) storing HITs in DNS AAAA entries and looking them up by domain name, 3) name resolution service for HITs such as OpenDHT [RFC6537], 4) an ad hoc HIT exchange service to populate files on each machine, or 5) support for DNS extensions described in RFC 5205.

Over time, support for these techniques has varied. The HIPL project has experimented with all of them. OpenHIP lacks support for option 2, and HIP4BSD lacks support for options 1 and 3.

Implementing opportunistic HIP mode in a clean way is challenging, as HITs need to be known when an application binds or connects to a socket. Approach 2 has been difficult in practice due to resistance of sysadmins to include AAAA entries for HITs in the DNS server, and is a non-standards-compliant use of the resource record. Approach 3 is being progressed with two independent implementations of a HIP-OpenDHT interface. At the moment, the easiest way for enabling experimentation appears to be approach 4 when a shell script based on Secure SHell (SSH) and Secure Copy (SCP) can connect to a peer machine and copy HITs to the local configuration files. However, this approach is not scalable or secure for the long run. HIPL developers have had positive experiences with alternative 5.

2.3.2. Opportunistic Mode

In opportunistic mode, the Initiator starts a base exchange without knowledge of the Responder's HIT. The main advantage of the opportunistic mode is that it does not require additional lookup infrastructure for HIs [RFC5205] [RFC6537].

The opportunistic mode also has a few disadvantages. First, the Initiator may not identify the Responder uniquely just based on the IP address in the presence of private address realms [RFC5770]. Second, the Initiator has to settle for a "leap of faith"; that is, assume there is no man-in-the-middle attack. However, this can be partially mitigated by using certificates at the Responder side [RFC6253] or by prompting the user using a graphical interface to explicitly accept the connection [paper.usable-security].

The opportunistic mode requires only minor changes in the state machine of the Responder and small changes for the Initiator [paper.leap-of-faith]. While the Responder can just select a suitable HIT upon receiving the first HIP base exchange packet (known as an "I1") without a predefined HIT for the Responder, the Initiator should be more careful in processing the first packet from the Responder, known as the "R1". For example, the Initiator should make sure that it can disambiguate simultaneously initiated opportunistic base exchanges from each other.

In the context of the HIPL project, the opportunistic mode has been successfully applied at the HIP layer for service registration [RFC5203]. HIP4BSD implemented opportunistic mode successfully with small modifications to the FreeBSD socket layer to support opportunistic mode. However, the Linux implementation was more challenging, as described below.

The HIPL project experimented with opportunistic mode by interposing a shim at two different layers. In the first approach, an API-based shim was implemented to capture socket calls from the application. This was somewhat complicated to implement and explicitly enabling an individual application (or groups of applications) to use the opportunistic mode was required. In the second approach [paper.leap-of-faith], the shim was placed between the network and transport layers. Upon successful base exchange, the shim translated IP-based packet flows to HIT-based packet flows by re-injecting the translated packets back to the networking stack.

Unless bypassed for DNS, both of the opportunistic mode implementation approaches in HIPL subjected the application(s) to undergo opportunistic mode procedures also for DNS requests. Both approaches also implemented an optional "fall back" to non-HIP base connectivity if the peer did not support HIP. The detection of peer support for HIP was based on timeouts. To avoid timeouts completely and to reduce the delay to a single Round-Trip Time (RTT) for TCP, the project also experimented with TCP-specific extensions [thesis.bishaj].

The OpenHIP project experimented with opportunistic mode through the use of an opportunistic (-o) option. For the Responder, this option determines whether or not HIP accepts I1s received with a zeroed receiver's HIT. On the Initiator's side, this option allows one to configure a name and LSI in the known Host Identities file. When the HIT field is missing, an I1 is sent with a zeroed receiver's HIT. The LSI is needed by an IPv4 application to trigger the association. Note that, normally, the LSI used is based on the bottom 24 bits of the HIT, but in the case of opportunistic mode, the HIT is unknown; thus, the LSI may differ from the HIT.

As a summary of the opportunistic mode experimentation, it is possibly best suited for HIP-aware applications. Either it can be used by HIP itself in registration extensions or by native HIP applications [RFC6317]. This way, the inherent security trade-offs of the opportunistic mode are explicitly visible to the user through the HIP-aware application.

2.3.3. Resolving HITs to Addresses

When HIP is used in opportunistic mode, the Initiator does not know the Responder's HIT, but it does know its IP address. In most other cases, however, the kernel or applications may know the HITs and not the IP addresses; in these cases, an IP address resolution step for HITs must take place.

A few techniques have been experimented with. First, OpenDHT can also use HITs as keys for IP address records. Second, work by Ponomarev has shown that the reverse DNS tree may be used for reverse lookups of the ORCHID space [HIT2IP]. Third, the need for resolution may trigger some type of HIP bootstrap message, similar in some sense to an Address Resolution Protocol (ARP) message (to resolve the HIT). The bootstrap (BOS) packet used to be present in the early revisions of the HIP base specifications, but it was removed from the final specifications due to insufficient interest at the time. The HIPL implementation currently sends an I1 to a link broadcast IP address if it doesn't know the IP address of the peer. It has triggered warnings in some Windows hosts running antivirus software that classified broadcasts with unknown protocol number as intrusion attempts. The utility of this technique is limited to the local link.

2.3.4. IPsec Management API Extensions

A generic key management API for IP security is known as PF_KEY API [RFC2367]. PK_KEY is a socket protocol family that can be used by trusted applications to access the IPsec key engine in the operating system. Users of this interface typically need sysadmin privileges.

HIP-related extensions to the PF_KEY interface define a new protocol IPPROTO_HIP. Their main functionality is replacing the TCP and UDP checksum with a HIP-compatible checksum (because the transport pseudoheader is based on HITs) in incoming and outgoing packets. Recent Linux kernel versions do not require patching for these extensions.

2.3.5. Transport Protocol Issues

When an application triggers a HIP base exchange through the transport protocol, the first data packet can be lost unless the HIP and IPsec implementation is able to buffer the packet until the base exchange completes and IPsec SAs are set up. The loss of the data packet when it is a TCP SYN packet results in TCP timeout [RFC6298] that unnecessarily delays the application. A loss of a UDP packet can cause even longer timeouts in applications. Therefore, it was found to be important for HIP implementations to support the

buffering of the packet. On the other hand, if the HIP base exchange or UPDATE takes longer than 1 second, which is the case on lightweight devices, a spurious timeout can occur at the transport layer. The HIP implementation could prevent this scenario by manipulating timeout values at the transport layer or, alternatively, dropping the original or retransmitted duplicate packet.

The multihoming support in [RFC5206] is intended for the purpose of failover, when a host starts using an alternative locator when a current locator fails. However, a host could used this multihoming support for load balancing across different locators. Multihoming in this manner could potentially cause issues with transport protocol congestion control and loss detection mechanisms. However, no experimental results from using HIP multihoming in this capacity have been reported.

The use of paths with different characteristics can also impact the estimate of a retransmission timer at the sender's transport layer. TCP uses a smoothed average of the path's Round-Trip Time and its variation as the estimate for a retransmission timeout. After the retransmission timer expires, the sender retransmits all outstanding packets in go-back-N fashion.

When multihoming is used for simultaneous data transmission from several locators, there can easily be scenarios when the retransmission timeout does not correspond to the actual value. When packets simply experience different RTT, its variation is high, which sets the retransmission timeout value unnecessarily high. When packets are lost, the sender waits excessively long before retransmitting. Fortunately, modern TCP implementations deploying Selective Acknowledgments (SACKs) and Limited Transmit are not relying on retransmission timeouts except when most outstanding packets are lost.

Load balancing among several paths requires some estimate of each path's capacity. The TCP congestion control algorithm assumes that all packets flow along the same path. To perform load balancing, the HIP layer can attempt to estimate parameters such as the delay, bandwidth, and loss rate of each path. A HIP scheduler could then distribute packets among the paths according to their capacity and delay, to maximize overall utilization and minimize reordering. The design of the scheduler is a topic of current research work; none are reported to exist. Different network paths can have different Maximum Transmission Unit (MTU) sizes. Transport protocols perform MTU discovery typically only in the beginning of a connection. As HIP hides mobility from the transport layer, it can happen that packets on the new path get fragmented without knowledge of the transport protocol. To solve this problem, the HIP layer could

inform the transport layer of mobility events. Protocols to support such notifications to the transport layer have been proposed to the IETF in the past, including transport triggers [TRIGTRAN], lightweight mobility detection and response (LMDR) [LMDR], and TCP response to connectivity change [TCP-RLCI].

2.3.6. Legacy NAT Traversal

Legacy NAT traversal for outbound-initiated connections to a publicly addressed Responder has been implemented by all three HIP implementations; two (HIPL and HIP4BSD) implement Interactive Connectivity Establishment (ICE) techniques [RFC5770] for inbound NAT traversal. It has also been reported that the use of Teredo [RFC4380] over HIP was simpler than the modifications required for ICE techniques because Teredo effectively manifests itself as a routable, virtual locator to the system. UDP encapsulation is now the default mode of HIP operation for OpenHIP's IPv4 HIP implementation. Finding an IPv6 NAT implementation for experiments has been difficult. In addition, the initial implementations of NAT traversal for HIP based on ICE techniques proved to be complicated to implement or integrate, and a native NAT traversal mode is now under development for HIP [NAT-TRAVERSAL]. NAT traversal is expected to be a major mode of HIP operation in the future.

2.3.7. Local Management of Host Identity Namespace

One issue not being addressed by some experimental implementations is how to perform source HIT selection across possibly multiple host identities (some may be unpublished). This is akin to source address selection for transport sockets. How much HIP policy to expose to users is a user interface issue. Default or automatic configuration guesses might have undesirable privacy implications for the user.

Helsinki University of Technology (TKK, now Aalto) has implemented an extension of the native HIP API to control multiple host identities [thesis.karlsson]. A problem with Linux routing and multiple identities was discovered by the HIPL development group. As Linux routing is based on longest prefix match, having multiple HITs on virtual devices is problematic from the viewpoint of access control because the stack selects the source HIT based on the destination HIT. A coarse-grained solution for this is to terminate the longest prefix match for ORCHIDs in the Linux networking statck. However, a more fine-grained solution tries to return a source HIT matching to the algorithm used for generating the destination HIT in order to facilitate compatibility with new algorithms standardized in the future.

There are security and privacy issues with storing private keys securely on a host. Current implementations simply store private keys in a file that is readable only by applications with root privileges. This may not be a sufficient level of protection, as keys could be read directly from the disk or, e.g., some application with a set-user-id flag. Keys may be stored on a trusted platform module (TPM), but there are no reported HIP experiments with such a configuration. In a Boeing pilot project, temporary certificates were generated from a key on a USB SIM chip and used in the HIP base exchange. Use of certificates in HIP requires extensions to the HIP specifications [RFC6253]. Another option is encrypting keys on disks and keeping a passkey in memory (like in Secure Socket Layer (SSL) certificates on servers, that ask for a password when booting Linux).

2.3.8. Interactions with Host Firewalls

HIP is presently an experimental protocol, and some default firewall configuration scripts on popular Linux distributions do not permit the HIP number. Determining which rules to modify without compromising other policies can be tricky; the default rule set on a previous SuSE Linux distribution was discovered to contain over one hundred rules. Moreover, it may be the case that the end user has no control over the firewall settings, if administered by an enterprise IT department. However, the use of HIP over UDP has alleviated some of these concerns. When using HIP over UDP, the firewall needs to allow outbound UDP packets and responses to them.

2.4. IPv4 versus IPv6 Issues

HIP has been oriented towards IPv6 deployment, but all implementations have also added support for IPv4. HIP supports IPv6 applications well, as the HITs are used from the general IPv6 address space using the ORCHID prefix. HITs are statistically unique, although they are not routable at the IP layer. Therefore, a mapping between HITs and routable IP addresses is necessary at the HIP layer, unless an overlay network or broadcast technique is available to route packets based on HITs.

For IPv4 applications, a 32-bit Local Scope Identifier (LSI) is necessary at the sockets API. The LSI is an alias for a host identity and is only meaningful within one host. Note that an IPv4 address may be used as an LSI if it is configured to refer to a particular host identity on a given host, or LSIs may be drawn from an unallocated IPv4 address range, but lack of coordination on the LSI space may hinder implementation portability.

HIP makes it possible to use IPv6 applications over the IPv4 network and vice versa. This has been called "interfamily operation" (flexibility between different address families) and is enabled by the fact that the transport pseudoheader is always based on HITs regardless of whether the application or the underlying network path is based on IPv4. All three open source HIP implementations have demonstrated some form of interfamily handoff support. The interfamily portion of the BEET patch in the Linux kernel was found more difficult to complete compared with the single-family processing.

HIP also provides the potential to perform cross-family support, whereby one side of a transport session is IPv6 based and another is IPv4 based [paper.handovers].

2.5. What Have Early Adopters Learned from Experience?

Implementing HIP in current stacks or as overlays on unmodified stacks has generally been successful. Below are some caveats and open issues.

Experimental results comparing a kernel versus user-space HIP implementations in terms of performance and DoS resilience would be useful. If the kernel implementation is shown to perform significantly better than the user-space implementation, it may be a sufficient justification to incorporate HIP within the kernel. However, experiences on general purpose laptops and servers suggests that for typical client use of HIP, user-space implementations perform adequately.

Although the HIPL kernel-based keying implementation was submitted to the Linux kernel development process, the implementation was not accepted. The kernel developers felt that since Mobile IP (MIP) and the Internet Key Exchange Protocol (IKE) are implemented as user-space signaling daemons in Linux, that should be the approach for HIP, too. Furthermore, the kernel patch was somewhat big, affecting the kernel in many places and having several databases. The Linux kernel maintainers did eventually accept the BEET patch.

Some users have been explicitly asking about the coexistence of HIP with other VPN and Mobile IP software. On Windows, VPN clients tend to install their own versions of TAP drivers that might conflict with the driver used by the OpenHIP implementation. There may also be issues due to lack of coordination leading to unintended HIP-over-VPN sessions or lack of coordination of the ESP Security Parameter Index (SPI) space. However, these types of conflicts are only speculation

and were not reported to the research group; only some positive reports of HIP and VPN software properly coexisting have been reported by the HIPL group.

With legacy applications, LSI support is important because IPv6 is not widely used in applications. The main issues in getting applications to work well over HIP have been related to bugs in the implementations themselves, or latency related issues (such as TCP timeouts due to Linux IPsec implementation). There have been no major obstacles encountered in practice, and there has also been some experience in using HIP with native applications [paper.p2psip].

3. Infrastructure Implications

This section focuses on the deployment of infrastructure to support HIP hosts.

3.1. Impact on DNS

HIP DNS extensions [RFC5205] were developed by NEC Eurolabs and contributed to OpenHIP and were also developed by the HIPL project, both for the BIND9 DNS server. Legacy applications do not query for HIP resource records, but DNS proxies (local resolvers) interpose themselves in the resolution path and can query for HI records. The BIND 9 deployment for HIPL uses binary blob format to store the HIP resource records; this means that no changes to the DNS server are required.

There have been no studies reported on the impact of changes based on [RFC5205] to HIP on the existing DNS. There have been some studies on using DNS to store HITs in the reverse tree [HIT2IP].

3.2. HIP-Aware Middleboxes

A design of a HIP registration protocol for architectured NATs (NATs that are HIP aware and use HIP identifiers to distinguish between hosts) has been completed and published as RFC 5204. Performance measurement results with a prototype are available, but experimentation on a wide scale is still missing. RFC 5207 provides a problem statement for HIP-aware NATs and middleboxes [RFC5207].

As argued by Aura, et al. [paper.hipanalysis], the encryption of the Initiator Host Identity (HI) prevents policy-based NAT and firewall support, and middlebox authentication, for HIP. The catch is that when the HI is encrypted, middleboxes in the network cannot verify the signature of the second base exchange packet from the Initiator

(I2) and, thus, cannot safely create a state for the HIP association. On the other hand, if the HI is not encrypted, a stateful middlebox can process the I2 and create protocol state for the session.

3.3. HIT Resolution Infrastructure

OpenDHT HIT-to-IP address resolution has been implemented by Aalborg University, Denmark, Helsinki Institute for Information Technology for HIPL, and by Boeing for OpenHIP [RFC6537].

The prototype of the Host Identity Indirection Infrastructure (Hi3) has been implemented using OpenHIP and HIPL. A set of 25 i3 servers was running on PlanetLab for several years. While a PlanetLab account is required to run the servers, anybody could openly use the provided service.

The main idea of Hi3 is to transmit HIP control packets using the i3 system as a lookup and rendezvous service, while transmitting data packets efficiently end-to-end using IPsec. Performance measurements were conducted comparing the association setup latency, throughput, and RTT of Hi3 with plain IP, HIP, and i3 [paper.hi3].

One difficulty has been with debugging the i3 system. In some cases, the messages did not traverse i3 correctly, due to its distributed nature and lack of tracing tools. Making the system work has been challenging. Further, since the original research work was done, the i3 servers have gone offline.

NATs and firewalls have been a major disturbance in Hi3 experiments. Many networks did not allow incoming UDP packets to go through, therefore, preventing messages from i3 servers to reach the client.

So far, the Hi3 system has been evaluated on a larger scale only analytically. The problem is that running a larger number of clients to create a sufficient load for the server is difficult. A cluster on the order of a hundred Linux servers is needed for this purpose. Contacts to a State Supercomputer Centre in Finland have not been successful so far. A possible option is to use one of the existing Emulab installations, e.g., in Utah, for these tests.

3.4. Rendezvous Servers

A rendezvous server (RVS) [RFC5204] has been implemented by HIIT for HIPL, and an implementation also exists for OpenHIP. The concept has been extended to a relay server in [RFC5770]. Initial experimentation with the HIPL implementation produced the following observations:

- RVS may be better than dynamic DNS updates for hosts that change their address rapidly.

- Registration of a HIP host to RVS costs a base exchange.

- UPDATE and CLOSE packets sent through rendezvous servers is advised; RVS handling of UPDATE messages can typically solve the double jump [MULTI-HOMED] mobility problem.

The following advanced concepts need further study:

- Multiple RVSs per host for fault tolerance (e.g., one rendezvous node crashes) and an algorithm for selecting the best RVS.

- Load balancing. An RVS server could distribute I1s to different Responders if the Responder's identity is shared or opportunistic HIP is used.

- Offering a rendezvous service in a P2P fashion by HIP hosts.

3.5. Hybrid DNS-DHT Resolution

In addition to pure DNS and pure DHT HIP name resolution, a hybrid approach combining the standard DNS interface for clients with last-hop DHT resolution was developed. The idea is that the benefits of DNS solution (wide deployment, support for legacy applications) could be combined with advantages of DHT (fault tolerance, efficiency in handling flat data keys). The DHT is typically run internally by the organization managing the last-hop DNS zone and the DNS server. That way, the HITs belonging to that organization could be stored locally by the organization that improves deployability of the resolution system. However, organizations could also share a DHT between themselves or connect their DNS servers to a publicly available DHT, such as OpenDHT. The benefit of running a DHT on a local server cluster compared to a geographically spread DHT is higher performance due to decreased internal DHT latencies.

The system was prototyped by modifying the BIND DNS server to redirect the queries for HITs to a DHT server. The interface was implemented in XML according to specifications [RFC6537]. The system is completely backward compatible to legacy applications since the standard DNS resolver interface is used.

Performance of the system was evaluated by performing a rapid sequence of requests for querying and updating the HIT-to-IP address mapping. The request rate was varied from 1 to 200 requests per second. The average latency of one query request was less than 50 ms and the secured updated latency less than 100 ms with a low request

rate. However, the delay was increasing exponentially with the request rate, reaching 1 second for 200 requests per second (update rate 0) and almost 2 seconds (update rate 0.5). Furthermore, the maximum delay exceeded the mean by several times.

Based on experiments, a multi-processor system could handle more than 1000 queries per second. The latencies are dominated by the DHT resolution delay, and the DNS component is rather small. This is explained by the relative inefficiency of used DHT implementation (Bamboo) and could be definitely improved in the future.

4. Application Implications

In a deployed HIP environment, applications may be HIP aware or HIP unaware. RFC 5338 [RFC5338] describes various techniques to allow HIP to support unmodified applications. Some additional application considerations are listed below.

4.1. Non-Intrusive HIP Insertion

One way to support legacy applications that use dynamic linking is to dynamically interpose a modified resolver library. Using HIPL, several legacy applications were shown to work without changes using dynamic re-linking of the resolver library. For example, the Firefox web browser successfully worked with an Apache web server. The re-linking just requires configuring an LD_PRELOAD system variable that can be performed in a user shell profile file or as a start-up wrapper for an application. This provides the user with fine-grained policy control over which applications use HIP, which could alternately be considered a benefit or a drawback depending on whether the user is burdened with such policy choices. The technique was also found to be sensitive to loading LD_PRELOAD twice, in which case the order of linking dynamic libraries must be coded carefully.

Another method for transparently using HIP, which has no reported implementation experience, is via local application proxies (e.g., squid web proxy) that are modified to be HIP aware. Discussion of proxies for HIP is a current focus of research group activities [HIPRG-PROXIES].

4.2. Referrals

A concern that FTP would not work due to the problem of application referrals, i.e., passing the IP address within application messages, was discovered not to be a problem for FTP in practice. It is shown to work well both in the passive and active modes [paper.namespace]. It remains an open question how big problem referrals really are in

the practice. At least, they do not seem used for the client side because they are behind NATs, and, therefore, client addresses are unsuitable as referrals.

4.3. Latency

Some applications may be sensitive to additional RTTs or processing due to HIP resolutions or the protocol itself. For instance, page load speed for web browsers is a critical metric for browser designers. Some applications or deployments may not wish to trade application speed for the security and mobility management that HIP offers.

5. Network Operator's Perspective

There is no known deployment of HIP by a data service provider. However, some issues regarding HIP have been brought to the HIP research group by a network provider and are summarized below and in [HIP-OPERATORS].

5.1. Management of the Host Identity Namespace

When a network operator deploys HIP for its customers, several issues with management of host identities arise. The operator may prefer to generate the host identity itself rather than let each host create the identities. Several factors can create such a need. Public-private key generation is a demanding operation that can take tens of seconds on a lightweight device, such as a mobile phone. After generating a host identity, the operator can immediately insert it into its own AAA databases and network firewalls. This way, the users would not need to be concerned with technical details of host identity management.

The operator may use a Public Key Infrastructure (PKI) to certify host identities of its customers. Then, it uses the private key of an operator's Certificate Authority (CA) to sign the public key of its customers. This way, third parties possessing the public key of the CA can verify the customer's host identity and use this information, e.g., for admission control to infrastructure. Such practice raises the security level of HIP when self-generated host identities are used.

When the operator is using neither PKI nor DNS Security (DNSSEC) host names, the problem of securely exchanging host identities remains. When HIP is used in opportunistic mode, a man-in-the-middle can still intercept the exchange and replace the host identities with its own.

For instance, the signaling provided by SIP could be used to deliver host identities if it were secured by existing mechanisms in the operator's network.

5.2. Use of ESP Encryption

The research group has discussed whether operators can provide "value-added" services and priority, and comply with wiretapping laws, if all sessions are encrypted. This is not so much a HIP issue as a general end-to-end encryption issue.

The processing power of mobile devices also must be considered. One study evaluated the use of HIP and ESP on lightweight devices (Nokia N770 Internet Tablets having 200 MHz processors) [paper.mobiarch]. The overhead of using ESP on such a platform was found to be tolerable, about 30% in terms of throughput. With a bulk TCP transfer over WiFi, transfer without HIP was producing 4.86 Mbps, while over ESP security associations set up by HIP it was 3.27 Mbps. A lightweight HIP base exchange for this purpose is being developed at the time of this writing [HIP-DEX].

It is also possible to use HIP in a NULL encryption configuration if one of SHA1 or MD5 authentication are used.

5.3. Access Control Lists Based on HITs

A firewall typically separates an organization's network from the rest of the Internet. An Access Control List (ACL) specifies packet forwarding policies in the firewall. Current firewalls can filter out packets based on IP addresses, transport protocol, and port values. These values are often unprotected in data packets and can be spoofed by an attacker. By trying out common well-known ports and a range of IP addresses, an attacker can often penetrate the firewall defenses.

Furthermore, legacy firewalls often disallow IPsec traffic and drop HIP control packets. HIP allows ACLs to be protected based on packet exchanges that may be authenticated by middleboxes. However, HITs are not aggregatable, so HIT-based ACLs may be longer in length (due to an inability to group hosts with a single entry) and harder to deal with by human users (due to the length of the HIT compared with an IPv4 or IPv6 prefix).

Additionally, operators would like to grant access to the clients from domains such as example.com regardless of their current locators or HITs. This is difficult without a forward confirmed reverse DNS to use for non-repudiation purposes.

5.4. Firewall Issues

Helsinki University of Technology (TKK, now Aalto) has implemented a HIP firewall based on Linux iptables [paper.firewall] that operates in user-space.

In general, firewalls can be stateless, filtering packets based only on the ACL, and stateful, following and remembering packet flows. Stateless firewalls are simple to implement but provide only coarse-grained protection. However, their performance can be efficient since packet processing requires little memory or CPU resources. A stateful firewall determines if a packet belongs to an existing flow or starts a new flow. A flow identifier combines information from several protocol headers to classify packets. A firewall removes the state when the flow terminates (e.g., a TCP connection is closed) or after a timeout. A firewall can drop suspicious packets that fail a checksum or contain sequence numbers outside of the current sliding window.

A transparent firewall does not require that hosts within the protected network register or even know of the existence of the firewall. An explicit firewall requires registration and authentication of the hosts.

A HIP-aware firewall operating in the middle identifies flows using HITs of communicating hosts, as well as SPI values and IP addresses. The firewall must link together the HIP base exchange and subsequent IPsec ESP data packets. During the base exchange, the firewall learns the SPI values from I2 and R2 packets. Then, the firewall only allows ESP packets with a known SPI value and arriving from the same IP address as during the base exchange. If the host changes its location and the IP address, the firewall, if still on the path, learns about the changes by following the mobility update packets.

It is possible to implement a stateless, end-host-based firewall to reuse existing higher-layer mechanisms such as access control lists in the system. In this mode of operation, HITs would be used in the access control lists, and while the base exchange might complete, ESP is not passed to the transport layer unless the HITs are allowed in the access control list.

A HIP host can register to an explicit firewall using the usual procedure [RFC5203]. The registration enables the host and the firewall to authenticate each other. In a common case, where the Initiator and Responder hosts are located behind different firewalls, the Initiator may need to first register with its own firewall, and afterward, with the Responder’s firewall.

Some researchers have suggested that a firewall for security-critical environments should get involved in the base exchange and UPDATE procedures with middlebox-injected echo requests. Otherwise, the firewall can be circumvented with replay attacks if there is a compromised node within the network that the firewall is trying to protect [HIP-MIDDLE].

6. User Privacy Issues

Using public keys for identifying hosts creates a privacy problem as third parties can determine the source host even if attached to a different location in the network. Various transactions of the host could be linked together if the host uses the same public key. Furthermore, using a static IP address also allows linking of transactions of the host. Multiplexing multiple hosts behind a single NAT or using short address leases from DHCP can reduce the problem of user tracking. However, IPv6 addresses could reduce the occurrence of NAT translation and cause additional privacy issues related to the use of Media Access Control (MAC) addresses in IPv6 address autoconfiguration. HIP does provide for the use of anonymous (unpublished) HITs in cases in which the Initiator prefers to remain anonymous, but the Responder must be willing to accept sessions from anonymous peers.

With mutual authentication, the HIP Initiator should not have to reveal its identity (public key) to either a passive adversary or an active attacker. The HIP Initiator can authenticate the Responder's R1 packet before encrypting its host identity with the Diffie-Hellman-generated keying material and sending it in the I2 packet. The authentication step upon receiving an R1 defeats the active attacker (impersonator) of the Responder, and the act of encrypting the identity defeats the passive adversary. Since the Responder sends its public key unencrypted in the first reply message (R1) to the Initiator, the Responder's identity will be revealed to third-party on-path eavesdroppers. However, if the Responder authenticates the Initiator and performs access controls before sending the R1, the Responder can avoid disclosing its public key to an active attacker.

DNS records can provide information combining host identity and location information, the host public key, and host IP address. Therefore, identity and location privacy are related and should be treated in an integrated approach. The goal of the BLIND is to provide a framework for identity and location privacy [paper.blind] [HIP-PRIVACY]. The identity protection is achieved by hiding the actual public keys from third parties so that only the trusted hosts can recognize the keys. Location privacy is achieved by integrating traffic forwarding with NAT translation and decoupling host identities from locators. The use of random IP and MAC addresses

also reduces the issue of location privacy shifting the focus to protecting host identifiers from third parties. This approach is, by its very nature, incompatible with middlebox authentication.

To prevent revealing the identity, the host public key and its hash (HIT) can be encrypted with a secret key known beforehand to both Initiator and Responder. However, this is a requirement that cannot be easily implemented in practice. The BLIND framework provides protection from active and passive attackers using a modified HIP base exchange. If the host avoids storing its public keys in the reverse DNS or DHT repository, the framework achieves full location and identity privacy.

An alternative approach to reducing privacy threats of persistent identifiers is to replace them with short-lived identifiers that are changed regularly to prevent user tracking. Furthermore, identifiers must be changed simultaneously at all protocol layers; otherwise, an adversary could still link the new identifier by looking at an identifier at another protocol layer that remained the same after the change. The HIP privacy architecture that simultaneously changes identifiers on MAC, IP, and HIP/IPsec layers was developed at Helsinki University of Technology (TKK, now Aalto) [thesis.takkinen]. HIP could be extended in the future to allow active sessions to migrate identities.

7. Experimental Basis of This Report

This report is derived from reported experiences and research results of early adopters, implementers, and research activities. In particular, a number of implementations have been in development since 2002 (Section 2).

One production-level deployment of HIP has been reported. Boeing has described how it uses HIP to build Layer 2 VPNs over untrusted wireless networks [HIPLS]. This use case is not a traditional end-host-based use of HIP, but rather, it is one that uses HIP-aware middleboxes to create ESP tunnels on-demand between provider-edge (PE) devices.

The InfraHIP II project is deploying HIP infrastructure (test servers, rendezvous and relay servers) in the public Internet.

The following is a possibly incomplete list of past and current research activities related to HIP.

- Boeing Research & Technology (J. Ahrenholz, O. Brewer, J. Fang, T. Henderson, D. Mattes, J. Meegan, R. Paine, S. Venema, OpenHIP implementation, Secure Mobile Architecture)

- NomadicLab, Ericsson (P. Jokela, P. Nikander, J. Melen. BSD HIP implementation)
- Helsinki Institute for Information Technology (HIIT) (A. Gurtov, M. Komu, A. Pathak, D. Beltrami. HIPL, legacy NAT traversal, firewall, i3, native API)
- Helsinki University of Technology (TKK, now Aalto) (Janne Lindqvist, Niklas Karlsson, Laura Takkinen, and Essi Vehmersalo. HIP security and firewalls, multiple identities, and privacy management)
- University of California, Berkeley (A. Joseph, HIP proxy implementation)
- Laboratory of Computer Architecture and Networks, Polytechnic School of University of Sao Paulo, Brazil (T. Carvalho, HIP measurements, Hi3)
- Telecom Italia (M. Morelli, comparing existing HIP implementations)
- NEC Heidelberg (L. Eggert, M. Esteban, V. Schmitt working on RVS implementation, DNS, NAT traversal)
- University of Hamburg-Harburg (M. Shanmugam, A. Nagarajan, HIP registration protocol)
- University of Tuebingen (K. Wehrle, T. Lebenslauf to work on Hi3 or HIP-OpenDHT)
- University of Parma (UNIPR), Department of Information Engineering Parma, Italy. (N. Fedotova, HIP for P2P)
- Siemens (H. Tschofenig, HIP middleboxes)
- Denmark (Aalborg University, Lars Roost, Gustav Haraldsson, Per Toft, HIP evaluation project, OpenDHT-HIP interface)
- Microsoft Research, Cambridge (T. Aura, HIP analysis)
- MIT (H. Balakrishnan. Delegation-Oriented Architecture)
- Huawei (D. Zhang, X. Xu, hierarchical HIP architecture, HIP proxy, key revocation)

8. Related Work on ID-Locator Split

This section briefly summarizes the related work on the ID-locator split with particular focus on recent IETF and IRTF activity. In the academic research community, several related proposals were explored prior to the founding of this research group, such as the Internet Indirection Infrastructure (i3) [paper.i3], IPNL [paper.layered], DataRouter [paper.datarouter], Network Pointers [paper.netpointers], FARA [paper.fara], and TRIAD [paper.triad].

The topic of whether a new namespace is needed for the Internet has been controversial. The Namespace Research Group (NSRG) at the IRTF was not able to reach consensus on the issue, nor even to publish a final report. Yet, there seems to be little disagreement that, for many scenarios, some level of indirection from network name to network location is essential or highly desirable to provide adequate service. Mobile IP [RFC6275] is one example that reuses an existing namespace for host naming. Since Mobile IP was finalized, many new variants to providing this indirection have been suggested. Even prior to Mobile IP, the IETF has published informational documents describing architectures separating network name and location, including the work of Jerome Saltzer [RFC1498] and Nimrod [RFC1992].

Most recently, there have been standardization and development efforts in the IETF and IRTF as follows:

- The Site Multihoming in IPv6 (multi6) WG documented the ways that multihoming is currently implemented in IPv4 networks and evaluated several approaches for advanced multihoming. The security threats and impact on transport protocols were covered during the evaluation. The work continued in another WG, Site Multihoming by IPv6 Intermediation (shim6), which is focusing on specifications of one selected approach [RFC5533]. Shim6 uses the approach of inserting a shim layer between the IP and the transport layers that hides effects of changes in the set of available addresses. The applications are using one active address that supports referrals. Shim6 relies on cryptographically generated IPv6 addresses to solve the address ownership problem. HIP and shim6 are architecturally similar and use a common format for control packets. HIP specifications define only simple multihoming scenarios leaving such important issues as interface selection untouched. Shim6 offers complementary functionality that can be reused in HIP [REAP4HIP]. The OpenHIP implementation integrates HIP and shim6 protocols in the same framework, with the goal of allowing HIP to reuse the shim6 failure detection protocol. Furthermore, HIP and shim6 socket APIs have been jointly designed [RFC6317] [RFC6316].

- The IRTF Routing Research Group (RRG) has explored a class of solutions to the global routing scalability problem that involve either separation of the existing IP address space into those used for identifiers and locators as in LISP [LISP] and Six/One Router [SIX-ONE] and those advocating a fuller separation of these roles including ILNP [ILNP] and RANGI [RANGI].

- The End-Middle-End research group considered the potential for an explicit signaling and policy control plane for middleboxes and endpoints [EME]; at a joint meeting at IETF 69, the HIP and EME research groups discussed whether the EME framework could help HIP with middlebox traversal.

- The IETF Multipath TCP working group is developing mechanisms to simultaneously use multiple paths in a regular TCP session. The MPTCP solution aims to solve the multihoming problem also addressed by HIP but by solving it for TCP specifically.

- The Unmanaged Internet Protocol bears several similarities to the HIP architecture, such as the focus on identifiers that are not centrally managed that are also based on a cryptographic hash of a node's public key [thesis.ford].

- Apple Back To My Mac service provides secure connections between hosts using IPsec between a pair of host identifiers. However, the host identifier is reported to be an IPv6 Unique Local Addressing (ULA) address rather than a HIP identifier [RFC6281].

Although the HIP research group has not formally tried to compare HIP with other ID-locator split approaches, such discussions have occurred on other lists such as the Routing research group mailing list, and a comparison of HIP's mobility management solution with other approaches was published in [MOBILITY-COMPARISON].

9. Security Considerations

This document is an informational survey of HIP-related research and experience. Space precludes a full accounting of all security issues associated with the approaches surveyed here, but the individually referenced documents may discuss security considerations for their respective protocol component. HIP security considerations for the base HIP protocol can be found in Section 8 of [RFC5201].

10. Acknowledgments

Miika Komu, Pekka Nikander, Ari Keranen, and Jeff Ahrenholz have provided helpful comments on earlier draft versions of this document. Miika Komu also contributed the section on opportunistic mode. We also thank Dacheng Zhang for contributions on hierarchical HIP architectures and the Crypto Forum Research Group (Adam Back and Paul Hoffman) for clarification of Diffie-Hellman privacy properties.

11. Informative References

[BEET-MODE] Nikander, P. and J. Melen, "A Bound End-to-End Tunnel (BEET) mode for ESP", Work in Progress, August 2008.

[EME] Francis, P., Guha, S., Brim, S., and M. Shore, "An EME Signaling Protocol Design", Work in Progress, April 2007.

[HIP-DEX] Moskowitz, R., "HIP Diet EXchange (DEX)", Work in Progress, March 2011.

[HIP-MIDDLE] Hummen, R., Heer, T., Wehrle, K., and M. Komu, "End-Host Authentication for HIP Middleboxes", Work in Progress, October 2011.

[HIP-OPERATORS] Dietz, T., Brunner, M., Papadoglou, N., Raptis, V., and K. Kypris, "Issues of HIP in an Operators Networks", Work in Progress, October 2005.

[HIP-PRIVACY] Zhang, D. and M. Komu, "An Extension of HIP Base Exchange to Support Identity Privacy", Work in Progress, July 2011.

[HIPLS] Henderson, T., Venema, S., and D. Mattes, "HIP-based Virtual Private LAN Service (HIPLS)", Work in Progress, September 2011.

[HIPRG-PROXIES] Zhang, D., Xu, X., Yao, J., and Z. Cao, "Investigation in HIP Proxies", Work in Progress, October 2011.

[HIT2IP] Ponomarev, O. and A. Gurtov, "Embedding Host Identity Tags Data in DNS", Work in Progress, July 2009.

[ILNP] Atkinson, R., "ILNP Concept of Operations", Work in Progress, July 2011.

[LISP] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol (LISP)", Work in Progress, November 2011.

[LMDR] Swami, Y., Le, K., and W. Eddy, "Lightweight Mobility Detection and Response (LMDR) Algorithm for TCP", Work in Progress, February 2006.

[MOBILITY-COMPARISON] Thaler, D., "A Comparison of IP Mobility-Related Protocols", Work in Progress, October 2006.

[MULTI-HOMED] Huitema, C., "Multi-homed TCP", Work in Progress, May 1995.

[NAT-TRAVERSAL] Keranen, A. and J. Melen, "Native NAT Traversal Mode for the Host Identity Protocol", Work in Progress, January 2011.

[RANGI] Xu, X., "Routing Architecture for the Next Generation Internet (RANGI)", Work in Progress, August 2010.

[REAP4HIP] Oliva, A. and M. Bagnulo, "Fault tolerance configurations for HIP multihoming", Work in Progress, July 2007.

[RFC1498] Saltzer, J., "On the Naming and Binding of Network Destinations", RFC 1498, August 1993.

[RFC1992] Castineyra, I., Chiappa, N., and M. Steenstrup, "The Nimrod Routing Architecture", RFC 1992, August 1996.

[RFC2367] McDonald, D., Metz, C., and B. Phan, "PF_KEY Key Management API, Version 2", RFC 2367, July 1998.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

[RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, February 2006.

[RFC4423] Moskowitz, R. and P. Nikander, "Host Identity Protocol (HIP) Architecture", RFC 4423, May 2006.

[RFC4843] Nikander, P., Laganier, J., and F. Dupont, "An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers (ORCHID)", RFC 4843, April 2007.

[RFC5201] Moskowitz, R., Nikander, P., Jokela, P., and T. Henderson, "Host Identity Protocol", RFC 5201, April 2008.

[RFC5202] Jokela, P., Moskowitz, R., and P. Nikander, "Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP)", RFC 5202, April 2008.

[RFC5203] Laganier, J., Koponen, T., and L. Eggert, "Host Identity Protocol (HIP) Registration Extension", RFC 5203, April 2008.

[RFC5204] Laganier, J. and L. Eggert, "Host Identity Protocol (HIP) Rendezvous Extension", RFC 5204, April 2008.

[RFC5205] Nikander, P. and J. Laganier, "Host Identity Protocol (HIP) Domain Name System (DNS) Extensions", RFC 5205, April 2008.

[RFC5206] Nikander, P., Henderson, T., Vogt, C., and J. Arkko, "End- Host Mobility and Multihoming with the Host Identity Protocol", RFC 5206, April 2008.

[RFC5207] Stiemerling, M., Quittek, J., and L. Eggert, "NAT and Firewall Traversal Issues of Host Identity Protocol (HIP) Communication", RFC 5207, April 2008.

[RFC5338] Henderson, T., Nikander, P., and M. Komu, "Using the Host Identity Protocol with Legacy Applications", RFC 5338, September 2008.

[RFC5533] Nordmark, E. and M. Bagnulo, "Shim6: Level 3 Multihoming Shim Protocol for IPv6", RFC 5533, June 2009.

[RFC5770] Komu, M., Henderson, T., Tschofenig, H., Melen, J., and A. Keranen, "Basic Host Identity Protocol (HIP) Extensions for Traversal of Network Address Translators", RFC 5770, April 2010.

[RFC6253] Heer, T. and S. Varjonen, "Host Identity Protocol Certificates", RFC 6253, May 2011.

[RFC6275] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, July 2011.

[RFC6281] Cheshire, S., Zhu, Z., Wakikawa, R., and L. Zhang, "Understanding Apple's Back to My Mac (BTMM) Service", RFC 6281, June 2011.

[RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, June 2011.

[RFC6316] Komu, M., Bagnulo, M., Slavov, K., and S. Sugimoto, "Sockets Application Program Interface (API) for Multihoming Shim", RFC 6316, July 2011.

[RFC6317] Komu, M. and T. Henderson, "Basic Socket Interface Extensions for the Host Identity Protocol (HIP)", RFC 6317, July 2011.

[RFC6537] Ahrenholz, J., "Host Identity Protocol Distributed Hash Table Interface", RFC 6537, February 2012.

[SIX-ONE] Vogt, C., "Six/One: A Solution for Routing and Addressing in IPv6", Work in Progress, October 2009.

[TCP-RLCI] Schuetz, S., Koutsianas, N., Eggert, L., Eddy, W., Swami, Y., and K. Le, "TCP Response to Lower-Layer Connectivity-Change Indications", Work in Progress, February 2008.

[TRIGTRAN] Dawkins, S., Williams, C., and A. Yegin, "Framework and Requirements for TRIGTRAN", Work in Progress, February 2003.

[book.gurtov] Gurtov, A., "Host Identity Protocol (HIP): Towards the Secure Mobile Internet", ISBN 978-0-470-99790-1, Wiley and Sons, (Hardcover, p 332), June 2008.

[paper.blind] Ylitalo, J. and P. Nikander, "BLIND: A complete identity protection framework for end-points", Proc. of the Twelfth International Workshop on Security Protoc ols, April 2004.

[paper.datarouter]
Touch, J. and V. Pingali, "DataRouter: A Network-Layer Service for Application-Layer Forwarding", Proceedings of International Workshop on Active Networks (IWAN), May 2003.

[paper.fara]
Clark, D., Braden, R., Falk, A., and V. Pingali, "FARA: Reorganizing the Addressing Architecture", Proceedings of ACM SIGCOMM FDNA Workshop, August 2003.

[paper.firewall]
Lindqvist, J., Vehmersalo, E., Komu, M., and J. Manner, "Enterprise Network Packet Filtering for Mobile Cryptographic Identities", International Journal of Handheld Computing Research (IJHCR), Volume 1, Issue 1, Pages 79-94, January 2010.

[paper.handovers]
Varjonen, S., Komu, M., and A. Gurtov, "Secure and Efficient IPv4/IPv6 Handovers Using Host-Based Identifier-Locator Split", Proceedings of the 17th International Conference Software, Telecommunications, and Computer Networks, September 2009.

[paper.hi3] Gurtov, A., Korzon, D., Lukyanenko, A., and P. Nikander, "Hi3: An Efficient and Secure Networking Architecture for Mobile Hosts", Computer communication, 31 (2008), p. 2457- 2467, <http://www.cs.helsinki.fi/u/gurtov/papers/comcom_hi3.pdf>.

[paper.hipanalysis]
Aura, T., Nagarajan, A., and A. Gurtov, "Analysis of the HIP Base Exchange Protocol", Proc. of the 10th Australasian Conference on Information Security and Privacy (ACISP), July 2005.

[paper.i3] Stoica, I., Adkins, D., Zhuang, S., Shenker, S., and S. Surana, "Internet Indirection Infrastructure (i3)", Proceedings of ACM SIGCOMM, August 2002.

[paper.layered]
Balakrishnan, H., Lakshminarayanan, K., Ratnasamy, S., Shenker, S., Stoica, I., and M. Walfish, "A Layered Naming Architecture for the Internet", Proceedings of ACM SIGCOMM, August 2004.

[paper.leap-of-faith]
Komu, M. and J. Lindqvist, "Leap-of-faith security is enough for IP mobility", Proceedings of the 6th IEEE Conference on Consumer Communications and Networking Conference (CCNC 09), 2009.

[paper.mobiarch]
Khurri, A., Vorobyeva, E., and A. Gurtov, "Performance of Host Identity Protocol on Lightweight Hardware", Proceedings of ACM MobiArch, August 2007.

[paper.namespace]
Komu, M., Tarkoma, S., Kangasharju, J., and A. Gurtov, "Applying a Cryptographic Namespace to Applications", Proc. of First International ACM Workshop on Dynamic Interconnection of Networks, September 2005.

[paper.netpointers]
Tschudin, C. and R. Gold, "Network pointers", ACM SIGCOMM Computer Communications Review, Vol. 33, Issue 1, January 2003.

[paper.p2psip]
Koskela, J., Heikkila, J., and A. Gurtov, "A secure P2P SIP system with SPAM prevention", ACM Mobile Computer Communications Review, July 2009.

[paper.triad]
Cheriton, D. and M. Gritter, "TRIAD: A New Next-Generation Internet Architecture", July 2000, <http://www-dsg.stanford.edu/triad/triad.ps.gz>.

[paper.usable-security]
Karvone, K., Komu, M., and A. Gurtov, "Usable Security Management with Host Identity Protocol", Proc. of the IEEE/ACS International Conference on Computer Systems and Applications, May 2009.

[thesis.bishaj]
Bishaj, B., "Efficient Leap of Faith Security with Host Identity Protocol", Master thesis, Helsinki University of Technology, June 2008.

[thesis.ford]
Ford, B., "UIA: A Global Connectivity Architecture for Mobile Personal Devices", Doctoral thesis, Massachusetts Institute of Technology, September 2008.

[thesis.karlsson]
Karlsson, N., "Enabling Multiple Host Identities on Linux", Master thesis, Helsinki University of Technology, September 2005.

[thesis.takkinen]
Takkinen, L., "Host Identity Protocol Privacy Management", Master thesis, March 2006, <http://www.tml.tkk.fi/~anttiyj/Laura-Privacy.pdf>.

Authors' Addresses

Thomas Henderson
The Boeing Company
P.O. Box 3707
Seattle, WA
USA

EMail: thomas.r.henderson@boeing.com

Andrei Gurtov
University of Oulu
Centre for Wireless Communications CWC
P.O. Box 4500
FI-90014 University of Oulu
Finland

EMail: gurtov@ee.oulu.fi