Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 21, 2008

T. Henderson
The Boeing Company
P. Nikander
Ericsson Research NomadicLab
M. Komu
Helsinki Institute for Information Technology

November 18, 2007

Using the Host Identity Protocol with Legacy Applications
draft-ietf-hip-applications-02

Status of this Memo

Copyright Notice

Abstract

This document is an informative overview of how legacy applications can be made to work with the Host Identity Protocol (HIP). HIP proposes to add a cryptographic name space for network stack names. From an application viewpoint, HIP-enabled systems support a new address family of host identifiers, but it may be a long time until such HIP-aware applications are widely deployed even if host systems are upgraded. This informational document discusses implementation and Application Programming Interface (API) issues relating to using HIP in situations in which the system is HIP-aware but the applications are not, and is intended to aid implementors and early adopters in thinking about and locally solving systems issues regarding the incremental deployment of HIP.

Table of Contents

1. Introduction

The Host Identity Protocol (HIP) [1] is an experimental effort in the IETF and IRTF to study a new public-key-based name space for use as host identifiers in Internet protocols. Fully deployed, the HIP architecture would permit applications to explicitly request the system to send packets to another host by expressing a location-independent name of the host when the system call to send packets is performed. However, there will be a transition period during which systems become HIP-enabled but applications are not. This informational document does not propose normative specification or even suggest that different HIP implementations use more uniform methods for legacy application support, but is intended instead to aid implementors and early adopters in thinking about and solving systems issues regarding the incremental deployment of HIP.

When applications and systems are both HIP-aware, the coordination between the application and the system can be straightforward. For example, using the terminology of the widely used sockets Application Programming Interface (API), the application can issue a system call to send packets to another host by naming it explicitly, and the system can perform the necessary name-to-address mapping to assign appropriate routable addresses to the packets. To enable this, a new address family for hosts could be defined, and additional API extensions could be defined (such as allowing IP addresses to be passed in the system call, along with the host name, as hints of where to initially try to reach the host).

This document does not define a native HIP API such as described above. Rather, this document is concerned with the scenario in which the application is not HIP-aware and a traditional IP-address-based API is used by the application. To use HIP in such a situation, there are a few basic possibilities: i) allow applications to use IP addresses as before, and provide a mapping from IP address to host identifier (and back to IP address) within the system, ii) take advantage of domain name resolution to provide the application with either an alias for the host identifier or (in the case of IPv6) the host identity tag (HIT) itself, and iii) support the use of HITs directly (without prior DNS resolution) in place of IPv6 addresses. This document describes several variations of the above strategies and points out tradeoffs with each approach.

When HITs are used (rather than IP addresses) as peer names at the system API level, they can provide a type of "channel binding" in that the Encapsulating Security Payload (ESP) association formed by HIP is cryptographically bound to the name (HIT) invoked by the calling application.

2. Terminology

Host Identity: An abstract concept applied to a computing platform.

Host Identifier (HI): A public key of an asymmetric key pair used as a name for a Host Identity. More details are available in [1].

Host Identity Tag (HIT): A 128-bit quantity composed with the hash of a Host Identity. More details are available in [2] and [1].

Local Scope Identifier (LSI): A 32- or 128-bit quantity locally representing the Host Identity at the IPv4 or IPv6 API.

Referral: An event when the application passes what it believes to be an IP address to another application instance on another host, within its application data stream. An example is the FTP PORT command.

Resolver: The system function used by applications to resolve domain names to IP addresses.

3. Approaches for supporting legacy applications

This section provides examples of how legacy applications, using legacy APIs, can operate on a HIP-enabled system and use HIP. The examples are organized by the name used by an application (or application user) to name the peer system: an IP address, a domain name, or a HIT. Finally, some local address management issues are discussed.

Applications use IP addresses as short-lived local handles, long-lived application associations, callbacks, referrals, and identity comparisons. Each of the below mechanisms has implications on these different uses of IP addresses by legacy applications.

3.1. Using IP addresses in applications

Consider the case in which an application issues a "connect(ip)" system call to set the default destination to a system named by address "ip", but for which we would like to enable HIP to protect the communications. Since the application or user can not indicate a desire to use HIP through the standard sockets API when IP addresses are used, the decision to invoke HIP must be done on the basis of host policy. For example, when an IPsec-based implementation of HIP is being used, a policy may be entered into the security policy database that mandates to use or try HIP based on a match on the source or destination IP address, or other factors. The mapping of IP address to host identifier may be implemented by modifying the host operating system or by wrapping the existing sockets API, such as in the TESLA approach [3].

There are a number of ways that HIP could be used in such a scenario.

Manual configuration:

Pre-existing SAs may be available due to previous administrative action, or a binding between an IP address and a HIT could be stored in a configuration file or database.

Opportunistically:

The system could send an I1 to the Responder with an empty value for Responder HIT.

Using DNS to map IP addresses to HIs:

> If the responder has host identifiers registered in the forward DNS zone and has a PTR record in the reverse zone, the Initiator could perform a reverse+forward lookup to learn the HIT associated with the address. Although the approach should work under normal circumstances, it has not been tested to verify that there are no recursion or bootstrapping issues, particularly if HIP is used to secure the connection to the DNS servers. Unless secured with DNS security extensions, the use of the reverse DNS map is subject to well-known security limitations (an attacker may cause an incorrect IP address to domain name binding to occur).

Using the opportunistic mode or using DNS in the above fashion can cause additional setup delays compared to using plain IP. For opportunistic mode, a host must wait to learn whether the peer is HIP-capable, although the delays may be mitigated in some implementations by sending initial packets (e.g., TCP SYN) in parallel to the HIP I1 packet. For DNS lookups, there are resolution latencies.

Solutions preserving the use of IP addresses in the applications have the benefit of better support for applications that use IP addresses for long-lived application associations, callbacks, and referrals, although it should be noted that applications are discouraged from using IP addresses in this manner due to the frequent presence of NATs [4] and Section 3.3, because the binding between host identifier and address is weak and not visible to the application or user. In fact, the semantics of the application's "connect(ip)" call may be interpreted as "connect me to the system reachable at IP address ip" but perhaps no stronger semantics than that. HIP can be used in this case to provide perfect forward secrecy and authentication, but not to strongly authenticate the peer at the onset of communications. DNS with security extensions (DNSSEC) [5] could be used to authenticate the bindings between IP address and host identifier, if the necessary DNSSEC records were available and trusted.

The legacy application is unaware of HIP and cannot therefore notify the user when the application uses HIP. However, the operating system can notify the user of the usage of HIP through a user agent. Further, it is possible for the user agent to name the network application that caused a HIP-related event. This way, the user is aware when he or she is using HIP even though the legacy network application is not.

Using IP addresses at the application layer may not provide the full potential benefits of HIP mobility support. It allows for mobility if the system is able to readdress long-lived, connected sockets upon

a HIP readdress event. However, as in current systems, mobility will break in the connectionless case, when an application caches the IP address and repeatedly calls sendto(), or in the case of TCP when the system later opens additional sockets to the same destination.

Section 4.1.6 of the base HIP protocol specification [1] states that implementations that learn of HIT-to-IP address bindings through the use of HIP opportunistic mode must not enforce those bindings on later communications sessions. This implies that when IP addresses are used by the applications, systems that attempt to opportunistically set up HIP must not assume that later sessions to the same address will communicate with the same host.

3.2. Using DNS to map domain names to HIs

In the previous section, it was pointed out that a HIP-enabled system might make use of DNS to transparently fetch host identifiers prior to the onset of communication. For applications that make use of DNS, the name resolution process is another opportunity to use HIP. If host identifiers are bound to domain names (with a trusted DNS), the following are possible:

Return HIP LSIs and HITs instead of IP addresses:

The system resolver could be configured to return a Local Scope Identifier (LSI) or HIT rather than an IP address, if HIP information is available in the DNS that binds a particular domain name to a host identifier, and otherwise to return an IP address as usual. The system can then maintain a mapping between LSI and host identifier and perform the appropriate conversion at the system call interface or below. The application uses the LSI or HIT as it would an IP address. This technique has been used in overlay networking experiments such as the Internet Indirection Infrastructure (i3).

Locally use a HIP-specific domain name prefix:

One drawback to spoofing the DNS resolution is that some applications actually may want to fetch IP addresses (e.g., diagnostic applications such as ping, or processes that generate system log files). One way to provide finer granularity on whether the resolver returns an IP address or an LSI is to distinguish by the presence of a domain name prefix. Specifically, if the application requests to resolve "HIP-www.example.com" (or some similar prefix string), then the system returns an LSI, while if the application requests to resolve

"www.example.com", IP address(es) are returned as usual. The use of a prefix rather than suffix is recommended, and the use of a string delimiter that is not a dot (".") is also recommended, to reduce the likelihood that such modified DNS names are mistakenly treated as names rooted at a new top-level domain.

Fetch HIP records transparently:

A third option would be for the system to opportunistically query for HIP records in parallel to other DNS resource records, and to temporarily cache the HITs returned with a DNS lookup, indexed by the IP addresses returned in the same entry, and pass the IP addresses up to the application as usual. If an application connects to one of those IP addresses within a short time after the lookup, initiate a base exchange using the cached HITs. The benefit is that this removes the uncertainty/delay associated with opportunistic HIP, because the DNS record suggests that the peer is HIP-capable.

Since the LSI or HIT is non-routable, a couple of potential hazards arise, in the case of referrals, callbacks, and long-lived application associations. First, applications that perform referrals may pass the LSI to another system that has no system context to resolve the LSI back to a host identifier or an IP address. Note that these are the same type of applications that will likely break if used over certain types of network address translators (NATs). Second, applications may cache the results of DNS queries for a long time, and it may be hard for a HIP system to determine when to perform garbage collection on the LSI bindings. However, when using HITs, the security of using the HITs for identity comparison may be stronger than in the case of using IP addresses.

It may be possible for an LSI or HIT to be routable or resolvable, either directly or through an overlay, in which case it would be preferable for applications to handle such names instead of IP addresses. However, such networks are out of scope of this document.

3.3. Connecting directly to a HIT

The previous two sections describe the use of IP addresses and LSIs as local handles to host identifiers. A third approach, for IPv6 applications, is to configure the application to connect directly to a HIT (e.g., "connect(HIT)" as a socket call). This scenario has stronger security semantics, because the application is asking the system to send packets specifically to the named peer system. HITs have been defined as Overlay Routable Cryptographic Hash Identifiers (ORCHIDs) such that they cannot be confused with routable IP

addresses; see [2].

This approach also has a few challenges. Using HITs can be more cumbersome for human users (due to the flat HIT name space) than using either IPv6 addresses or domain names, Another challenge with this approach is in actually finding the IP addresses to use, based on the HIT. Some type of HIT resolution service would be needed in this case. A third challenge of this approach is in supporting callbacks and referrals to possibly non-HIP-aware hosts. However, since most communications in this case would likely be to other HIP-aware hosts (else the initial HIP associations would fail to establish), the resulting referral problem may be that the peer host supports HIP but is not able to perform HIT resolution for some reason.

3.4. Local address management

The previous sections focused mainly on client behavior (HIP initiator). We must also consider the behavior for servers. Typically, a server binds to a wildcard IP address and well-known port. In the case of HIP use with legacy server implementations, there are again a few options. As in Section 3.1 above, the system may be configured manually to always, optionally (depending on the client behavior), or never use HIP with a particular service, as a matter of policy, when the server specifies a wildcard (IP) address.

When a system API call such as getaddrinfo [6] is used for resolving local addresses, it may also return HITs or LSIs, if the system has assigned HITs or LSIs to internal virtual interfaces (common in many HIP implementations). The application may use such identifiers as addresses in subsequent socket calls.

In the case when resolvers can return multiple destination identifiers for an application, it may be configured that some of the identifiers can be HIP-based identifiers, and the rest can be IPv4 or IPv6 addresses. The system resolver may return HIP-based identifiers in front of the list of identifiers when the underlying system and policies support HIP. An application processing the identifiers sequentially will then first try a HIP-based connection and only then other non-HIP based connections. However, certain applications may launch the connections in parallel. In such a case, the non-HIP connections may succeed before HIP connections. Based on local system policies, a system may disallow such behaviour and return only HIP-based identifiers when they are found from DNS.

Some applications may try to bind a socket to a specific local address, or may implement server-side access control lists based on socket calls such as getsockname() and getpeername() in the C-based

socket APIs. If the local address specified is an IP address, again, the underlying system may be configured to still use HIP. If the local address specified is a HIT (Section 3.3), the system should enforce that connections to the local application can only arrive to the specified HIT. If a system has many HITs, an application that binds to a single HIT cannot accept connections to the other HITs in the system.

When a host has multiple HIs and the socket behavior does not prescribe the use of any particular HI as a local identifier, it is a matter of local policy as to how to select a HI to serve as a local identifier. However, systems that bind to a wildcard may face problems when multiple HITs or LSIs are defined. These problems are not specific to HIP per se, but are also encountered in non-HIP multihoming scenarios with applications not designed for multihoming.

As an example, consider a client application that sends an UDP datagram to a server that is bound to a wildcard. The server application receives the packet using recvfrom() and sends a response using sendto(). The problem here is that sendto() may actually use a different server HIT than the client assumes. The client will drop the response packet when the client implements access control on the UDP socket (e.g. using connect()).

Reimplementing the server application using the sendmsg() and recvmsg() to support multihoming (particularly considering the anchillary data) would be the ultimate solution to this problem, but with legacy applications is not an option. As a workaround, we make suggestion for servers providing UDP-based services with non-multihoming capable services. Such servers should announce only the HIT that matches to the default outgoing HIT of the host to avoid such problems.

Finally, some applications may create a connection to a local HIT. In such a case, the local system may use NULL encryption to avoid unnecessary encryption overhead, and may be otherwise more permissive than usual such as excluding authentication, Diffie-Hellman exchange, and puzzle.

4. Security Considerations

In this section we discuss the security of the system in general terms, outlining some of the security properties. However, this section is not intended to provide a complete risk analysis. Such an analysis would, in any case, be dependent on the actual application using HIP, and is therefore considered out of scope.

The three outlined scenarios differ considerably in their security properties. There are further differences related to whether DNSSEC is used or not, and whether the DNSSEC zones are considered trustworthy enough. Here we mean that the delegation chain from the reverse IP root should be trusted (typical trust anchor issues), and the DNS zone administrators in charge of the netblock should be trusted to put in the right information.

When IP addresses are used to represent the peer system, the security properties depend on the the configuration method. With manual configuration, the security of the system is comparable to a non-HIP system with similar IPsec policies. The security semantics of an initial opportunistic key exchange are roughly equal to non-secured IP; the exchange is vulnerable to man-in-the-middle attacks. However, the system is less vulnerable to connection hijacking attacks. If the DNS is used, if both zones are secured (or the HITs are stored in the reverse DNS record) and the client trusts the DNSSEC signatures, the system may provide a fairly high security level. However, much depends on the details of the implementation, the security and administrative practices used when signing the DNS zones, and other factors.

Using the forward DNS to map a domain name into an LSI is a case that is closest to the most typical use scenarios today. If DNSSEC is used, the result is fairly similar to the current use of certificates with TLS. If DNSSEC is not used, the result is fairly similar to the current use of plain IP, with the exception that HIP provides protection against connection hijacking attacks.

If the application is basing its operations on HITs, the connections become automatically secured due to the implicit channel bindings in HIP. That is, when the application makes a connect(HIT) system call, the resulting packets will either be sent to a node possessing the corresponding private key or the security association will fail to be established.

When the system provides (spoofs) LSIs or HITs instead of IP addresses as the result of name resolution, the resultant fields may inadvertently show up in user interfaces and system logs, which may cause operational concerns for some network administrators.

5. IANA Considerations

This document has no actions for IANA.

6. Acknowledgments

Jeff Ahrenholz, Gonzalo Camarillo, Alberto Garcia, Teemu Koponen, Julien Laganier, and Jukka Ylitalo have provided comments on different versions of this draft. Erik Nordmark provided the taxonomy of how applications use IP addresses in a previously expired Internet Draft. The document received substantial and useful comments during the review phase from David Black, Pekka Savola, Lars Eggert, and the DNS directorate.

7. Informative References

[1] Moskowitz, R., Nikander, P., Jokela, P., and T. Henderson, "Host Identity Protocol", draft-ietf-hip-base-10 (work in progress), October 2007.

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

[3] Salz, J., Balakrishnan, H., and A. Snoeren, "TESLA: A Transparent, Extensible Session-Layer Architecture for End-to-end Network Services", Proceedings of USENIX Symposium on Internet Technologies and Systems (USITS), pages 211-224, March 2003.

[4] Carpenter, B., "Architectural Principles of the Internet", RFC 1958, June 1996.

[5] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

[6] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.

Appendix A. Changes from previous versions

This section is to be removed by the RFC Editor before publication. It summarizes resolution of issues raised in the following reviews: (1) IESG last call, (2) Gen-ART review, and (3) DNS directorate review. Mobility and secdir reviews did not result in actionable comments.

A.1. From version-01 to version-02 (current)

Better clarity in the abstract and introduction about the goal of the draft; namely, that it is informational to help implementors and early adopters think about and solve deployment issues (comment from Pekka Savola).

Delete the second paragraph of 3 about the general applicability of replacing IP addresses with LSIs and HITs at the socket layer. (comment from Pekka Savola).

Delete comments in Section 3.2 on routable LSIs, as this is seen to be out of scope and potentially controversial or incomplete (comment from David Black).

Delete reference to Erik Nordmark's shim6 application referral draft, since it is a dead draft (comment from David Black). Instead, Erik is cited in the acknowledgments section for providing the taxonomy of IP address usage scenarios.

Clarify (and reference the base spec) in Sec. 3.1 that use of the opportunistic mode requires that systems not enforce that the HIT-to-IP address bindings learned will pertain to subsequent sessions to that IP address.

Section 3.2 drew comments from several reviewers. First, David Black raised the issue that spoofing IP addresses with HITs or LSIs raises risks that it may turn up in log records; this has been noted in the text. The section on using a DNS suffix to signal the preferred use of HIP was objected to by members of the DNS directorate and others (including the co-author Pekka Nikander), due to concern that queries to a new TLD might leak out. The current draft instead recommends a DNS prefix instead of suffix, due to a suggestion by Thomas Narten.

In section 3.1, clarify recursion issues that may arise when doing reverse+forward lookup of HIP records from DNS (comment from Pekka Savola).

Clarify more specifically in security considerations section the DNSSEC trust assumptions or security considerations (outline of text

provided by Pekka Savola, and similar comment raised by Peter Koch).

Clarified in security considerations section that IP address spoofing could cause some operational difficulties if they unexpectedly show up in log files or UIs (comment from David Black).

Clarified in Sec. 3.1 that opportunistic and DNS techniques can incur additional latency when compared to plain IP (comment from Lars Eggert)

Added third option to Section 3.2 for using DNS (transparently fetching HIP resource records when doing other RR queries), suggested by Lars Eggert and also by Olaf Kolkman.

Incorporated last-call comments from Miika Komu, which were all handled in Section 3.4: i) clarify multihoming issue for servers with multiple HITs, when receiving UDP, ii) clarify a problem that might arise for applications that do parallel connect, and iii) suggest that loopback HIP connections could use a NULL encryption.

Removed expired references and updated active references.

Incorporated additional review comments from Miika Komu, and some suggested replacement text, and added him as a co-author.

Authors' Addresses

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

Email: thomas.r.henderson@boeing.com

Pekka Nikander
Ericsson Research NomadicLab
JORVAS FIN-02420
FINLAND

Phone: +358 9 299 1
Email: pekka.nikander@nomadiclab.com

Miika Komu
Helsinki Institute for Information Technology
Metsaenneidonkuja 4
Helsinki FIN-02420
FINLAND

Phone: +358503841531
Email: miika@iki.fi

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).