Network Working Group
Request for Comments: 5338
Category: Informational

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

September 2008

Using the Host Identity Protocol with Legacy Applications

Status of This Memo

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) [RFC5201] 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 and users to explicitly request the system to send packets to another host by expressing a location-independent unique name of a peer host when the system call to connect or 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.

The discussion so far assumes that applications are written directly to a sockets API. However, many applications are built on top of middleware that exports a higher-level API to the application. In this case, for the purpose of this document, we refer to the combination of the middleware and the middleware-based application as an overall application, or client of the sockets API.

When HIP is enabled on a system, but the applications are not HIP-aware, there are a few basic possibilities to use HIP, each of which may or may not be supported by a given HIP implementation. We report here on techniques that have been used or considered by experimental HIP implementations. We organize the discussion around the policy chosen to use or expose HIP to the applications. The first option is that users are completely unaware of HIP, or are unable to control whether or not HIP is invoked, but rather the system chooses to enable HIP for some or all sessions based on policy. The second option is that the user makes a decision to try to use HIP by conveying this information somehow within the constraints of the unmodified application. We discuss both of these use cases in detail below.

HIP was designed to work with unmodified applications, to ease incremental deployment. For instance, the HIT is the same size as the IPv6 address, and the design thinking was that, during initial experiments and transition periods, the HITs could substitute in data structures where IPv6 addresses were expected. However, to a varying degree depending on the mechanism employed, such use of HIP can alter the semantics of what is considered to be an IP address by applications. Applications use IP addresses as short-lived local handles, long-lived application associations, callbacks, referrals, and identity comparisons [APP-REF]. The transition techniques described below have implications on these different uses of IP addresses by legacy applications, and we will try to clarify these implications in the below discussions.

2. Terminology

Callback: The application at one end retrieves the IP address of the peer and uses that to later communicate "back" to the peer. An example is the FTP PORT command.

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 [RFC5201].

Host Identity Tag (HIT): A 128-bit quantity composed with the hash of a Host Identity. More details are available in [RFC4843] and [RFC5201].

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

Long-lived application associations: The IP address is retained by the application for several instances of communication.

Referral: In an application with more than two parties, party B takes the IP address of party A and passes that to party C. After this, party C uses the IP address to communicate with A.

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

Short-lived local handle: The IP addresses is never retained by the application. The only usage is for the application to pass it from the DNS APIs (e.g., getaddrinfo()) and the API to the protocol stack (e.g., connect() or sendto()).

3. Enabling HIP Transparently within the System

When both users and applications are unaware of HIP, but the host administrator chooses to use HIP between hosts, a few options are possible. The first basic option is to perform a mapping of the application-provided IP address to a host identifier within the stack. The second option, if DNS is used, is to interpose a local agent in the DNS resolution process and to return to the application a HIT or a locally scoped handle, formatted like an IP address.

3.1. Applying HIP to Cases in Which IP Addresses Are Used

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 the host administrator would like to enable HIP to protect the communications. The user or application intends for the system to communicate with the host reachable at that IP address. 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 to try HIP based on a match on the source or destination IP address, port numbers, 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 [TESLA].

There are a number of ways that HIP could be configured by the host administrator in such a scenario.

Manual configuration:

Pre-existing Security Associations (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. Discussion of the security implications of the use or absence of DNS Security (DNSSEC) is deferred to the Security Considerations section.

Using HIP 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 and waiting some time to receive a HIP R1 before processing a TCP SYN/ACK. Note that there presently does not exist specification for how to invoke such connections in parallel. Resolution latencies may also be incurred when using DNS in the above fashion.

A possible way to reduce the above-noted latencies, in the case that the application uses DNS, 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, the host should 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.

3.2. Interposing a HIP-Aware Agent in the DNS Resolution

In the previous section, it was noted that a HIP-unaware application might typically use the DNS to fetch IP addresses prior to invoking socket calls. A HIP-enabled system might make use of DNS to transparently fetch host identifiers for such domain names prior to the onset of communication.

A system with a local DNS agent could alternately return a Local Scope Identifier (LSI) or HIT rather than an IP address, if HIP information is available in the DNS or other directory 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) and by at least one HIP implementation.

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 behavior and return only HIP-based identifiers when they are found from DNS.

If the application obtains LSIs or HITs that it treats as IP addresses, a few potential hazards arise. 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.

Finally, applications may generate log files, and administrators or other consumers of these log files may become confused to find LSIs or HITs instead of IP addresses. Therefore, it is recommended that the HIP software logs the HITs, LSIs (if applicable), corresponding IP addresses, and Fully Qualified Domain Name (FQDN)-related information so that administrators can correlate other logs with HIP identifiers.

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. Discussion

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 [RFC1958]. However, they have weaker security properties than the approaches outlined in Section 3.2 and Section 4, 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.

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 [RFC5201] 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.

The legacy application is unaware of HIP and therefore cannot 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. Based on usability tests from initial deployments, displaying the HITs and LSIs should be avoided in user interfaces. Instead, traditional security measures (lock pictures, colored address bars) should be used where possible.

One drawback to spoofing the DNS resolution is that some applications, or selected instances of an application, actually may want to fetch IP addresses (e.g., diagnostic applications such as ping). One way to provide finer granularity on whether the resolver returns an IP address or an LSI is to have the user form a modified domain name when he or she wants to invoke HIP. This leads us to consider, in the next section, use cases for which the end user explicitly and selectively chooses to enable HIP.

4. Users Invoking HIP with a Legacy Application

The previous section described approaches for configuring HIP for legacy applications that did not necessarily involve the user. However, there may be cases in which a legacy application user wants to use HIP for a given application instance by signaling to the HIP-enabled system in some way. If the application user interface or configuration file accepts IP addresses, there may be an opportunity to provide a HIT or an LSI in its place. Furthermore, if the application uses DNS, a user may provide a specially crafted domain name to signal to the resolver to fetch HIP records and to signal to the system to use HIP. We describe both of these approaches below.

4.1. Connecting to a HIT or LSI

Section 3.2 above describes the use of HITs or LSIs as spoofed return values of the DNS resolution process. A similar approach that is more explicit 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 [RFC4843].

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.

4.2. Using a Modified DNS Name

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. Limits of domain name length or label length (255 or 63, respectively) should be considered when prepending any prefixes.

4.3. Other Techniques

Alternatives to using a modified DNS name that have been experimented with include the following. Command-line tools or tools with a graphical user interface (GUI) can be provided by the system to allow a user to set the policy on which applications use HIP. Another common technique, for dynamically linked applications, is to dynamically link the application to a modified library that wraps the system calls and interposes HIP layer communications on them; this can be invoked by the user by running commands through a special shell, for example.

5. Local Address Management

The previous two sections focused mainly on controlling 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. 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 [RFC3493] 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.

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 4), the system should enforce that connections to the local application can only arrive to the specified HIT. If a system has many HIs, an application that binds to a single HIT cannot accept connections to the other HIs but the one corresponding to the specified HIT.

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 a 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 ancillary 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 or public key 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.

6. 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 scenarios outlined above differ considerably in their security properties. When the DNS is used, there are further differences related to whether or not DNSSEC [RFC4033] is used, and whether the DNS zones are considered trustworthy enough. Here we mean that there should exist a delegation chain to whatever trust anchors are available in the respective trees, and the DNS zone administrators in charge of the netblock should be trusted to put in the right information.

When IP addresses are used by applications to name the peer system, the security properties depend on 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 Transport Layer Security (TLS). If DNSSEC is not used, the result is fairly similar to the current use of plain IP, with the additional protection of data integrity, confidentiality, and prevention of connection hijacking that opportunistic HIP provides. If DNSSEC is used, data integrity and data origin authentication services are added to the normal DNS query protocol, thereby providing more certainty that the desired host is being contacted, if the DNS records themselves are trustworthy.

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, either the resulting packets will 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. Therefore, it is recommended that the HIP software logs the HITs, LSIs (if applicable), corresponding IP addresses, and FQDN-related information so that administrators can correlate other logs with HIP identifiers.

7. Acknowledgments

Jeff Ahrenholz, Gonzalo Camarillo, Alberto Garcia, Teemu Koponen, Julien Laganier, and Jukka Ylitalo have provided comments on different versions of this document. The document received substantial and useful comments during the review phase from David Black, Lars Eggert, Peter Koch, Thomas Narten, and Pekka Savola.

8. Informative References

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

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

[TESLA] 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.

[RFC1958] Carpenter, B., Ed., "Architectural Principles of the Internet", RFC 1958, June 1996.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

[RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.

[APP_REF] Nordmark, E., "Shim6 Application Referral Issues", Work in Progress, July 2005.

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