Leap-of-Faith Security is Enough for IP Mobility

Miika Komu
Helsinki Institute for Information Technology
Helsinki University of Technology and University of Helsinki
Email: miika.komu@iki.fi

Janne Lindqvist
Helsinki University of Technology
Department of Computer Science and Engineering
Email: janne.lindqvist@tml.hut.fi

Abstract— **Host mobility presents a challenge for security protocols. For example, many proposals exist for integrating IPsec to Mobile IP. However, the existing approaches are cumbersome to configure and contain many round trips for security and mobility updates. The Host Identity Protocol (HIP) is being developed in the IETF to provide secure host mobility and multihoming. The default way to operate the protocol is that the connection initiator knows the peer's public key or a hash of the public key. This requires either infrastructure support or pre-configuration which introduces difficulties for deploying the protocol. In this paper, we present an implementation and evaluation of HIP that creates leap-of-faith security associations. The implemented approach establishes end-to-end security without requiring any new infrastructure to be deployed. We argue that since worldwide PKI is nowhere near, and seems to nearly impossible to br deploy in practice, leap-of-faith security is enough for Internet access and mobility. In our view, the deployment of opportunistic HIP even makes the deployment of DNSSEC unnecessary for most applications.**

I. Introduction

In the vast number of approaches to host mobility, many of the proposals ignore security issues. For example, extensively researched Mobile IP(v6) protocol introduces difficulties with IPsec [1].

Host Identity Protocol [2] integrates to IPsec to secure mobility and multihoming. In the HIP architecture, the IP addresses are relieved from their role as identifiers by public keys or hashes of the public keys. When the IP address of the host changes, the connection is still bound the the same cryptographically secure identity. Thus, transport layer connections can tolerate changes in IP addresses using HIP.

In Mobile IPv6, the server side does not need any changes, but it can support mobility optimizations. The security of Mobile IPv6 was designed to avoid introducing any new security threats to the Internet [3]. Mobile IP uses IP address as the identifier. On the other hand, Host Identity Protocol was designed to introduce a new cryptographic identity space for Internet and to use IPsec as the default mechanism to protect transport layer communication.

The concept of leap-of-faith security or weak authentication between untrusted principals [4] has been used and implemented in many security protocols. For example, Secure Shell (SSH) protocol uses leap-of-faith security as as follows. When a client connects to a server the first time, the user sees and verifies the fingerprint of the server's public key, and the SSH software stores the public key to disk. Next time the client connects to the server, the SSH client software compares the server key to the one stored on the disk. If they do not match, SSH alerts the user of a possible man-in-the-middle attack. Thus, the assumption of the leap-of-faith security is that there is no active attacker in the network during the first connection.

In this paper, we present the design and implementation of a leap-of-faith security approach to HIP called the opportunistic mode. The opportunistic mode is briefly described in the base specification of HIP, but, for example, API issues are left aside. The literature does not contain any experimental results on opportunistic mode and therefore we have experimented with a way of of implementing the opportunistic mode and its APIs which do not interfere with the normal operation of HIP. The implementation supports incremental deployment because it allows fallback to non-HIP based communication when the peer does not support HIP. We argue that the opportunistic end-to-end security approach is enough for Internet access for heterogeneous wired and wireless networks since the deployment of global public-key infrastructure is virtually impossible. It should be noted that AAA architectures are beyond the scope of this paper, since we focus on end-to-end security. We have implemented the approach with HIP, but the same experiences should be applicable to other end-to-end mobility protocols.

The rest of the paper is organized as follows. We first proceed to introduce the Host Identity Protocol architecture. Then, we discuss related work. Next, we give the design and implementation details, followed by discussion section. We finish the article in conclusions.

II. Host Identity Protocol

Host Identity Protocol (HIP) [2] introduces a new global namespace, the Host Identifier (HI) namespace, for transport-layer connections. The namespace separates transport layer and network layer locators. This allows transport layer connections to survive when the network-layer address changes due to end-host mob5Bility or multihoming. The new global namespace makes it also possible to name and contact hosts behind private-address realms controlled by NAT boxes [5].

The HIP namespace is unmanaged in the sense that no central authority for creating the names exists. A name in the namespace is statistically unique. Namespace collisions are highly unlikely due to the size of the addresses space which corresponds to the IPv6 address space. The namespace is cryptographic by its nature; a HI is essentially a public key.

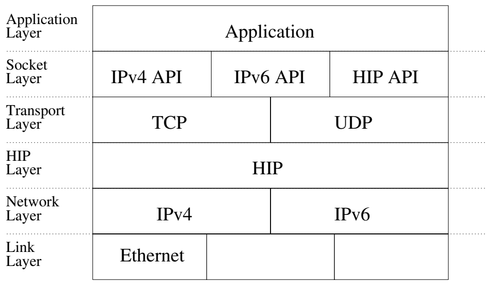

Fig. 1. HIP layering and naming architecture

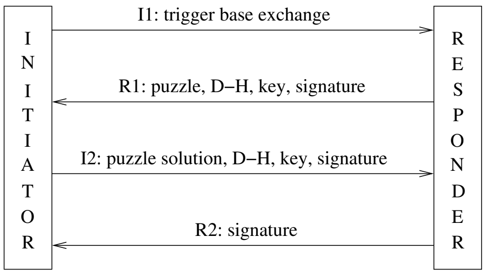
INITIATOR
RESPONDER
I1: trigger base exchange
R1: puzzle, D−H, key, signature
I2: puzzle solution, D−H, key, signature
R2: signature

Fig. 2. HIP base exchange

Forging of such names is very difficult. In addition, HIs can be used for strong end-to-end and end-to-middle authentication.

Public keys can be of variable length and longer keys cannot be used as socket endpoints in the existing sockets API [6]. To be able to pass public keys through the current sockets API with existing applications and networking stacks, HIP employs two shorter, fixed-size presentations of public keys. The shorter representations are also used in the protocol messages to ease protocol encoding and to reduce the length of control messages. First, Host Identity Tag (HIT) is a hash of the public key. The HIT has a fixed 28 bit prefix to separate it from routable IPv6 addresses. Second, Local Scope Identifier (LSI) is an identifier that can fit into an IPv4 address to support IPv4-only applications and is valid only in the context of the local host [7].

The HIP namespace requires a new layer in the networking stack to handle translation between HIs and routable addresses (i.e. locators). The new layer is located between transport and network layers, and it translates HIs to locators and vice versa. The HIP layer can map a HI either to an IPv4 or IPv6-based locator dynamically as illustrated in Figure 1.

Typically, a user inputs the client application the server host name and service name. Then, the system resolver translates these names to their machine readable representations. Effectively, the resolver searches the host name from DNS and returns the corresponding IP address(es) and numerical service port number. The address information may also also the HITs of the server if they were found from DNS. When the application connects to a HIT, this triggers a base exchange between the hosts. Upon successful completion, IPsec ESP [8] secures the application data between the client and server.

HIP uses IPsec Encapsulated Security Payload (ESP) to secure and encrypt communication between two hosts [8]. However, before IPsec can be used, the hosts need to create shared secret keys with each other. The procedure to create the keys in HIP is called the base exchange [9] and it is illustrated in Figure 2. The base exchange is a four way Diffie-Hellman key exchange during which hosts negotiate the IPsec keys and algorithms, and learn each others public keys. All base exchange messages, except the first one, are signed with DSA or RSA private keys to protect the integrity of the messages. The initiator starts the base exchange with an *I1* message that does not contain any signature to prevent DoS attacks that try to trick the responder to consume its time in verifying signatures from unauthenticated initiators. Normally, the I1 message would contain the responder's HIT (HIT(R)), but in opportunistic mode, the HIT(R) field is empty. Responder replies with an *R1* message that contains always the responder's HIT. It includes also responder's public key and a computational puzzle. The puzzle protects the responder from denial of service attacks because the responder can increase its difficulty when it is under attack. The initiator solves the puzzle and responds with a signed I2 message that contains also the initiator's public key. The responder receives the *I2* and validates the puzzle and other parameters, and concludes the base exchange with the *R2*. After this, both hosts have authenticated each other and have established keys for ESP protected application data transfer.

HITs are problematic from the viewpoint of DNS which supports only hierarchical identifiers. HITs are flat in the sense that they do not contain any hierarchical information. For this reason, resolving a HIT to a hostname or IP address using the currently deployed DNS is impossible. There are at least two scenarios where this can be a problem. First, a problem can occur e.g. when a server receives a connection from a HIT of a client and tries to verify the HIT from DNS. An example case of this are IRC servers that typically try to check the client information with a reverse DNS query and force the IRC connection to timeout when the check fails. In the second case, which is referred to as the *referral problem*, an application can communicate the HIT of its peer application to a third application running on some other host. The host of the third application may not have any means to route packets to the HIT because the host cannot resolve the HIT to an IP address using DNS. For example, FTP supports this kind of behaviour [6]. In such a case, it is possible to use an overlay to store and retrieve HIT-to-IP mappings, or use the overlay to route the packets without any IP address information at the end-hosts [10].

Basic mobility in HIP is based on a moving host always informing its peers on its new location using a signed message. Each peer then verifies that the message is from the authentic host and authentic location before sending any ESP traffic to the new location. During the verification procedure, which is called as the return routability test, the peer effectively sends a signed nonce to the host in the new location. Then, the host in the new location signs the nonce with its own public key and sends the nonce back. This way, the hosts can authenticate handovers and the return routability test protects against replay attacks.

III. Related Work

In this section, we describe different leap-of-faith security approaches that, however, do not provide global Internet-wide mobility and multihoming.

Koponen et al. [11] have proposed suspend-mode mobility for SSH and TLS. Their approach is based on the assumption that suspend and resume activity for laptops is the usual case for mobility, and that host mobility in Mobile IP and HIP models is not needed. As a counter argument, it is perceivable that deployment of heterogeneous networks necessitates mobility as supported by HIP or Mobile IP. Koponen et al's approach is applicable only to SSH-based connections. In contrast, our approach supports both unmodified legacy applications and also UDP-based network communication.

RFC4322 [12] describes opportunistic encryption for IKE. The idea in the RFC is to distribute public keys in the DNS and use DNSSEC [13] to protect against active attacks [12]. If DNSSEC is deployed, our opportunistic security approach can also leverage it. A benefit of the approach is that it does not require new Resource Record fields to the DNS. A drawback is that the approach does not support mobility and multihoming.

An approach for implementing early opportunistic key agreement for ad hoc networks is described by Candolin et al in [14]. The approach is based is to utilize ICMP to establish IPsec security associations that can be used to secure e.g. neighbor discovery. The drawback of the approach is that it does not support global host mobility.

IV. Implementation

This section presents the implementation design of our leap-of-faith security approach and performance tests.

A. Design

We have implemented opportunistic mode in HIP for Linux implementation. We implemented the opportunistic mode as an interposition library that intercepts socket calls and translates IP addresses to HITs as shown in Figure 3. The library supports legacy applications because it does not require any modifications to the source code of applications.

A developer, system administrator or user can use the library using two methods. In the first method, a developer links the source code of an invidual application to the library. The second method, dynamic linking, is easier for administrators and users, and does not require access to the source code. Dynamic linking means that the user or administrator enables the library dynamically using `LD_PRELOAD` environment variable. This can be done with different granularities, such as per application, per user or per system, depending on security requirements. The different granularities can also be used to avoid the overhead of introduced by security. To reduce the configuration steps for users, the opportunistic mode should be provided at system level. Both client and server-based

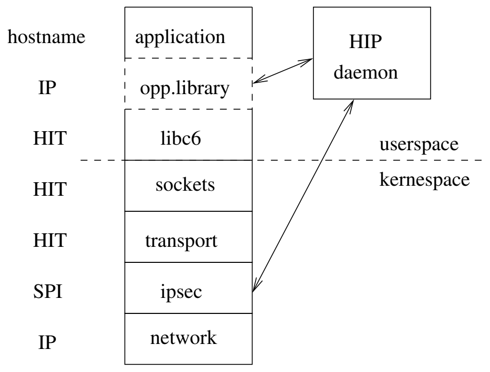

Fig. 3. Layering and Software Module Organization

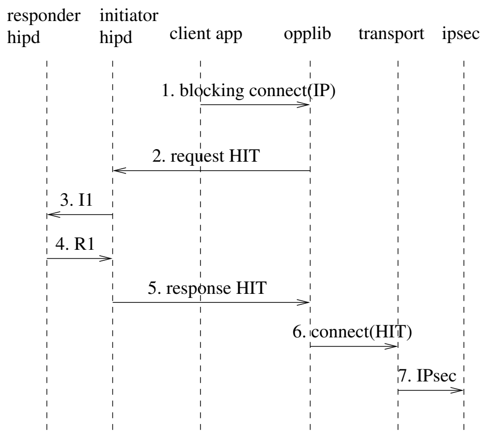

Fig. 4. Flow Diagram of Opportunistic Base Exchange

applications can use the library, although we believe that it is more beneficial to client applications.

The process of the opportunistic handshake is illustrated in Figure 4 from the viewpoint of an initiator. In step 1, the application calls a socket API function that sends data using IP addresses, for example connect() or sendto(). The opportunistic library then intercepts the function call. In step 2, the library (denoted as opplib in the figure) queries the HIP daemon for the corresponding HIT. The query blocks until the HIP daemon responds. The daemon triggers opportunistic base exchange with the peer in step 3. Upon receiving the R1 in step 4, the HIP daemon sends the responder's HIT to the library in step 5 and proceeds with the base exchange. Now, the opportunistic library can proceed with the translation and connect to the HIT of the responder in step 6. Finally, the control flow proceeds from transport to IPsec layer processing in step 7, which transmits the data over ESP.

The library consists of roughly 2000 physical lines (SLOC) of C code. It does not translate raw sockets or sockets that are already bound to HITs and it can translate both IPv4 and IPv6 addresses to HITs. It creates a completely new HIT-based socket for each IP-based socket because the sizes of IPv4 and IPv6 addresses are different.

The library processes datagram-oriented (UDP) socket calls differently from connection-oriented (TCP) socket calls. With datagrams, the local or peer address can change at any point of the communication and may require a new HIP base exchange. To avoid unnecessary base exchanges, the library caches the address-to-HIT translation for a specific socket until the application changes the peer address and only then triggers a new opportunistic base exchange to discover the HIT of the peer.

The opportunistic mode was mostly implemented within the library, but it also required some changes to the existing HIP daemon. The changes in daemon were minimal to implement the responder processing. Namely, the responder just selects a local HIT for R1 when the destination HIT of the I1 is NULL.

The initiator part of the daemon required to store state information related to opportunistic connections in the HIP association database. The database stores the state of HIP base exchanges and it is indexed using HITs of the initiator and responder. The I1 packet has an empty destination HIT in opportunistic mode and when the daemon accesses the database for such a packet, it does not use a blank destination HIT to index it. Instead, the daemon calculates a "pseudo HIT" based on the HIT prefix and the IP address of the responder. This avoids mixing up multiple simultaneous opportunistic base exchanges and protects against active attackers that try to send R1 packets to the initiator from other network addresses. It should be noticed that the daemon skips the pseudo HIT generation if it finds existing HIP association with the peer and returns the peer HIT immediately.

The implementation supports fallback to plain TCP/IP based on timeouts when an initiator detects that the peer does not support HIP. The daemon resumes the blocked library when it does not receive an R1 during a certain period of time. The library then discards the translation step and proceeds with unprotected communication using the original IP address. The implementation caches the addresses of the peers that did not support HIP to avoid triggering the base exchange unnecessarily again. The daemon flushes cache entries upon two events. First, this occurs when the HIP association closes or expires. Second, this occurs when local or peer host moves. Local host movement causes flushing because it might move between NATted realms that have overlapping private address spaces. Peer host movement address causes flushing because the previous IP address of the peer could be occupied by a new host and new connections to this IP should be directed to the host that presently occupies the address.

B. *Performance*

We evaluated the performance of the opportunistic mode on two 64-bit 2 GHz Intel Dual Core computers running Ubuntu 8.04. The machines were connected to each other using a direct 1 Gbit link. We used 1024-bit RSA asymmetric keys as HIs. We employed 128-bit AES keys for HIP and ESP encryption and 160-bit SHA1 keys for IPsec authentication. The linux kernel version of the computers were 2.6.25.8 and included BEET [8] patches for ESP that were adopted as part of the standard linux kernel in an upstream kernel version. The standard deviations were relatively small the measurements and therefore they are not included in this section.

We observed an average 47 ms delay for an application complete TCP connect() call to a HIT, which consists of the time to complete the base exchange and the TCP handshake. The delay was roughly the same with the opportunistic library. However, we experienced a minimum of three 3 seconds with each LSI measurement. The reason for this is that the LSI implementation does not yet cache data packets that the host sends during the base exchange. Instead, it just drops the packets. The Linux TCP stack includes fixed three second retransmission timeout by default when the first TCP packet is lost. The same behavior does not occur in the current library implementation because it blocks the socket calls and therefore no packets are actually lost.

The RTT between the two machines was 0.261 ms with plain ICMPv4, 0.319 ms using HITs and 0.658 ms with LSIs. We did not measure RTT with the opportunistic library because it does not support translation of raw sockets yet. Iperf version 2.0.2-4 showed a throughput of 942 Mbits/s for plain TCP, 300 Mbits/s for HIT-based TCP connection, 296 MBits/s for the opportunistic HIP library and 94 MBits/s for LSI-based connection. We assume that the LSI throughput was so low because the implementation is highly unoptimized.

We tested the opportunistic library in a mobility scenario where a TCP-based test application running on a local host created a connection to a peer application residing on a remote host. Then, we forced the local host to change its IP address to a new one while the TCP connection was still active. As a result, the TCP connection survived because of the HIP handover and local application was still able to send data to the peer application. The connection survived even though the local application was bound to the old IP address because the library was still translating the old address to the HIT.

V. Discussion

Our performance measurements indicate that the library offer the same performance as HIT based data transfers. The library exceeded the performance of the LSI implementation because it is still very unoptimized.

We observed that an opportunistic base exchange was typically invoked twice for the first connection of an application. The reason for this was that all of the library calls were wrapped, including also the DNS query. The DNS query triggered the first base exchange and the connection to the peer application triggered the second. This could be avoided by caching the IP addresses in the implementation, or by setting up by-pass policies for DNS ports.

The library requires further experimentation with varying kind of applications and environments. We have successfully tested it with the Firefox web browser, iperf and some other simple applications that create network connections in simple fashion. We are in the process of experimenting the library with a wider range of networking applications, such as real-time applications.

We are still improving the library implementation to meet some additional challenges. For example, we would like the library to support non-blocking operation. Also, the library does not yet cover all possible socket calls even though it handles the most common ones. We have also experienced that LD_PRELOAD was difficult to apply in some systems, such as CentOS 5.2 and Maemo tablets. For such systems, we suggest to apply the opportunistic mode as explained in [15].

Opportunistic HIP requires a relaxed security model in terms of leap of faith or time. It makes two security-related assumptions. The first assumption is that a remote host cannot guess the time when the initiator sends the I1 and reply using its own R1. The second assumption is that an attacker can snoop the trigger but cannot reply using its own R1. Typically, packet snooping is quite easy in e.g. wireless networks which makes the opportunistic mode especially vulnerable in such kind of broadcast environment. However, this quite difficult in practice because the attacker has to forge its address as the destination address of the I1 trigger and to be able to respond from this address. The opportunistic mode is a "better than nothing" security guarantee until sufficiently amount of HIP infrastructure is deployed on the Internet. We believe that heterogeneous wired and wireless networks cannot support a global PKI in any case.

The fallback approach, however, could introduce the possibility to attack the base exchange by down-negotiation. The attacker (middleman) can just drop the I1 packet. The initiator now sends e.g. a TCP SYN to the Recipient and the middleman catches it, and establishes connections to the initiator and the responder. This way, the middleman does not need to use any cryptographic operations to catch the traffic. However, this attack can be mitigated by security policies in the end-hosts, for example, by not allowing unencrypted connections.

In addition to attackers, packet loss could also be a problem with the fallback approach. The problem with the fallback is that it is based on timeouts. The timeout has to be long enough to provide reasonable guarantee against packet losses, but, on the other hand, long timeouts frustrate the user. Long timeouts could be avoided using explicit detection of HIP capability of the peer in a backwards compatible way as described in more detail in [16].

There are different pros and cons in using different kind of identifiers at the application layer, we do not see HITs, LSIs and opportunistic IP identifiers as rivalling or conflicting approaches. On the contrary, they complement each other to support different kinds of applications. However, a system should provide some kind of default policy for selection among different kinds of identifiers. For the default policy, we suggest that the application first try to resolve HIs of the peer. This is required for forwards compatibility with PKI. When the application supports IPv6 through the system resolver, the resolver returns the HIT of the peer to the application. When the application supports only IPv4, the resolver returns an LSI instead. This way, the application can detect the presence of HIP. It then tries the opportunistic mode as the last option, i.e., when the resolver returns just an IP address to the application in the absence of a HI. For backwards compatibility, the opportunistic library should fall back to non-HIP communication when the peer does not respond with an R1 during certain period of time.

VI. Conclusions

We have presented the design and implementation of a leap-of-faith end-to-end security architecture for host mobility. Our implementation allows legacy applications to establish IPsec security associations and change their points of network attachment without losing connectivity. One of the major advantages of our architecture is the backward compatible implementation of leap-of-faith security as the hosts can fall back to plain TCP or UDP connections when a peer does not support HIP. Hosts benefit from secure mobility, assuming that an active attacker cannot mount a man-in-the-middle attack in the beginning of the communication. We showed that our approach is sufficient to secure communication in heterogeneous network environments in the lack of a global PKI or DNSSEC deployment.

References

[1] V. Devaparalli and F. Dupont, "RFC 4877: Mobile IPv6 Operation with IKEv2 and the revised IPsec Architecture," Apr. 2007.

[2] R. Moskowitz and P. Nikander, "Host Identity Protocol Architecture," IETF, RFC 4423, May 2006.

[3] T. Aura and M. Roe, "Designing the Mobile IPv6 Security Protocol," *Annales des télécommunications / Annals of telecommunications, special issue on Network and information systems security*, no. 3-4, March-April 2006.

[4] J. Arkko and P. Nikander, "How to authenticate unknown principals without trusted parties," in *LNCS 2845: Security Protocols, 10th International Workshop*, Apr. 2003.

[5] M. Komu, T. Henderson, P. Matthews, H. Tschofenig, and A. Keränen, "Basic HIP extensions for traversal of network address translators draft-ietf-hip-nat-traversal-04," July 2008, work in progress, expires in Jan 2009.

[6] M. Komu, S. Tarkoma, J. Kangasharju, and A. Gurtov, "Applying a Cryptographic Namespace to Applications," in *Proc. of the first ACM workshop on Dynamic Interconnection of Networks (DIN 2005)*. Cologne, Germany: ACM Press, Sept. 2005.

[7] T. Henderson, P. Nikander, and M. Komu, *RFC 5338: Using the Host Identity Protocol with Legacy Applications*, Sept. 2008.

[8] P. Jokela, R. Moskowitz, and P. Nikander, "Rfc5202: Using the encapsulating security payload (ESP) transport format with the host identity protocol (HIP)," IETF, RFC 5202, Apr. 2008.

[9] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson, "RFC5201: Host identity protocol," Apr. 2008.

[10] P. Nikander, J. Arkko, and B. Ohlman, "Host identity indirection infrastructure (Hi3)," in *Proc. of The Second Swedish National Computer Networking Workshop 2004 (SNCNW2004)*, Karlstad, Sweden, Nov. 2004.

[11] T. Koponen, P. Eronen, and M. Särelä, "Resilient Connections for SSH and TLS," in *USENIX Annual Technical Conference*, May 2006.

[12] M. C. Richardson and D. H. Redelmeier, "RFC 4322: Opportunistic Encryption using the Internet Key Exchange (IKE)," Dec. 2005.

[13] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "RFC 4033: DNS Security Introduction and Requirements," IETF," RFC, Mar. 2005.

[14] C. Candolin, J. Lundberg, and P. Nikander, "Experimenting with early opportunistic key agreement," in *Proceedings of Workshop SEcurity of Communication on Internet, Internet Communication Security*, Sept. 2002.

[15] T. Finéz, "Backwards compatibility experimentation with host identity protocol and legacy software and networks," Master's thesis, Helsinki University of Technology, Department of Computer Science, 2008.

[16] B. Bishaj, "Efficient leap of faith security with host identity protocol," Master's thesis, Helsinki University of Technology, Department of Computer Science, June 2008.