Implementing Host Identity Protocol

Miika Komu <miika@iki.fi>
Helsinki Institute for Information Technology
17.11.2008
Why Are We Doing HIP?

• The cleanest patch to fix several flaws in the Internet in an architecturally coherent way
  – Mobility and multihoming for transport layer
  – Transport layer confidentiality and integrity protect.
  – End-to-end NAT traversal
  – Works both with TCP and UDP for legacy apps!
  – Works with IPv4 and IPv6 apps and networks!

• Customers: Researchers, P2P-SIP and PISA

• It's fun!
Host Identity Protocol for Linux (HIPL)

- Linux-oriented, open source implementation of HIP
  - Nokia Tablets are also supported
  - Partial Symbian support
- Supports several protocol and impl. extensions
  - Base exchange, mobility, RVS, NAT, HIP proxy
  - Supports both kernel and userspace ipsec
- Two other active projects
  - Ericsson's BSD-oriented implementation
  - Boeing's OpenHIP (linux, windows, mac os x)
HIPL Implementation History 1/3

- Started as a student project in 2001 (four students)
- Continued 2002 in HIIT in Fuego-Core, InfraHIP and InfraHIP II projects by two of the students
- Implementation efforts and interoperability tests detailed provided feedback to the IETF drafts
  - Interoperability tests with IndraNet, Ericsson and Boeing
- Active participation to IETF standardization
HIPL Implementation History 2/3

- Started as kernelspace-oriented implementation
  - Asymmetric crypto was done using a userspace daemon
  - BEET was implemented as a hack to Linux IPsec
- Ported asymmetric crypto to the linux kernel
  - Nowadays there is RSA support in linux kernel
- Moved everything to userspace
  - Linux networking maintainers rejected our huge kernel patch
HIPL Implementation History 3/3

- **BEET patch**
  - Interfamily support and $\frac{1}{2}$ of BEET patch in 2.6.19
  - 2.6.27 contains rest of the BEET patch

- **HIP implementation has been moving from an research prototype towards an open source product**
  - Release 1.0.4 coming soon
  - Ubuntu and Fedora repositories
HIPL Implementation Architecture

- GUI notifies user for new host associations
- HIP daemon implements HIP control plane and controls IPsec
- Libraries / DNS proxy look-up HITs and convert HITs to IP addresses
- Both GUI and firewall can block connections
- Multiple fw extensions
HIT-based Connection Example

1. `getaddrinfo(hostname)`

2. `hostname`

3. `<HIT, IP>`

4. `<HIT, IP>`

5. 4. `<HIT, IP>`

6. `HIT`

7. `connect(HIT)`

8. `base exchange`

9. `ESP protected application data`

Application

Resolver or DNS Proxy

DNS

Socket Layer

Transport

HIP

IPsec

Network

Peer Host
Opportunistic Mode 1/3

- How to support HIP without (DNS) look-up infrastructure support in early HIP deployments?
  - Opportunistic mode establishes a connection to an unknown HIT
- What id to use in connect(id) and sendto(id) calls in opportunistic mode?
  - Alternative 1: “pseudo-HIT”
  - Alternative 2: IP address (implemented)
Opportunistic Mode 2/3

hostname

IP

HIT

SPI

IP

application

opp.library

libc6

sockets

transport

ipsec

network

HIP daemon

userspace

kernelspace
Opportunistic Mode 3/3

- Opportunistic mode hack: I1 is a TCP option
- Benefit: faster fallback to TCP/IP when peer does not support HIP
- Drawback: works only for TCP, not UDP
- Supported by the implementation already as an implementation extension
## Native APIs for HIP

<table>
<thead>
<tr>
<th>Layer</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Layer</td>
<td></td>
</tr>
<tr>
<td>Socket Layer</td>
<td>IPv4 API</td>
</tr>
<tr>
<td>Transport Layer</td>
<td>TCP</td>
</tr>
<tr>
<td>HIP Layer</td>
<td>HIP</td>
</tr>
<tr>
<td>Network Layer</td>
<td>IPv4</td>
</tr>
<tr>
<td>Link Layer</td>
<td>Ethernet</td>
</tr>
</tbody>
</table>
TLS Differences to IPsec

• Benefits
  – TLS has wider deployment (HTTPS)
  – TLS-over-TCP passes through NAT/FWs
  – TLS-over-TCP has automatic MTU discovery

• Drawbacks
  – TLS does protect the TCP port numbers
  – TLS-over-TCP is more prone to e.g. RST attacks
  – Separate protocol for UDP (DTLS)

• TLS requires to modify the application
  – TLS tunneling possible, but cumbersome
  – Both a burden and also the key to TLS success?
BTNS APIs

App #1  App #2  App #3  App #4

TLS/GSS/SASL

IPv4 and IPv6 APIs

SCTP  TCP  UDP

IPsec

IPv4  IPv6

Ethernet  Etc

IPv4 Layer

Transport Layer

Socket Layer

Appl. Layer

IPv4 and IPv6 APIs

TLS/GSS/SASL

App #1  App #2  App #3  App #4

Link Layer

Network Layer

IPsec Layer

Socket Layer

Appl. Layer
**DNS vs. OpenDHT**

- DNS is quite rigid and difficult to configure
  - Flat names and DHT (see Ponomarev's work)
- Distributed Hash Tables (DHTs) are currently better for flat names
  - Currently using Bamboo DHT
  - IPv6 patch for Bamboo (Lu Xiaopeng)
  - OpenDHT network unstable; deploying own DHT
- DNS seems to be a better long-term alternative
HIP NAT Traversal using ICE

1. base exchange with locators
2. pair up locators
3. connectivity tests
4. ESP

Initiator

Responder

HIP Relay Server
NAT Traversal using Teredo

• Teredo vs. ICE
  – Plenty of free Teredo servers available
  – Teredo requires an IPv6 application (+socket opt), but the “magic” happens outside the application
  – ICE is more intrusive for the application because requires changing the protocol semantics

• NAT traversal with HIP
  – HIP-ICE: ICE changes hidden within HIP software
  – HIP-Teredo: no changes to the HIP software
Protocol State in HIP

- Base exchange (mirrored state machine)
  - Initiator has to create state
  - R1 packets are stateless (i.e. fixed memory requirements)
  - RVS and NAT Relay are stateless towards Initiator
  - Firewalls and other HIP-aware middleboxes may add nonces to the HIP control messages as a security measure

- Mobility updates (asymmetric state machine)
  - Mobile host sends its current set of locators its peers
  - Peers verify the locators for reachability
  - RVS/Relay required for double jump
Mobility Management

- Locator management
  - Locators in base exchange
  - Source locator selection for UPDATE with locator
  - Interfamily handovers
- Symmetric vs. asymmetric routes
- Handovers with long disconnectivity create problems with TCP timeouts
  - TCP user timeout option
- Simultaneous multiaccess
  - Load balancing: connection vs. packet based
  - Which outbound security association to use?
Misc Implementation Fun

- Retransmissions
  - Different mechanism for base exchange and update
  - Choosing optimal retransmission timeout can be tricky (slow ADSL lines, slow WLAN authentication)
- HIP loopback
- Broadcasting of I1s
- Simultaneous initiators
- Userspace IPsec
Questions?

Miika Komu <miika@iki.fi>

Documentation and implementation at:

http://infrahip.hiit.fi/
References 1/2

- RFC5201, Host Identity Protocol, Jokela et al, Apr 2008
- RFC5206: End-host Mobility and Multihoming with Host Identity Protocol, Henderson at al, Apr 2008
- Integrating Mobility, Multi-homing and Security in a HIP way, Pekka Nikander et al, Feb 2003
- RFC5205: Host Identity Protocol Domain Name System Extension, Laganier at al, Apr 2008
References 2/2

- Native Application Programming Interfaces for Host Identity Protocol, Komu et al, Jul 2008
- Overview and Comparison Criteria for Host Identity Protocol and Related Technologies, Koponen et al, Feb 2005
- Leap-of-faith security is enough for IP mobility, Komu et al, Jan 2009
- Enterprise Network Packet Filtering for Mobile Cryptographic Identities, Janne Lindqvist et al, June 2007
- HIP Extensions for the Traversal of Network Address Traversal, Keränen et al, Oct 2008