End-Host Authentication for HIP Middleboxes

draft-heer-hip-middle-auth-01

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79. This document may not be modified, and derivative works of it may not be created, except to publish it as an RFC and to translate it into languages other than English.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on January 8, 2009.

Copyright Notice

Copyright (C) The IETF Trust (2008).

Abstract

The Host Identity Protocol [RFC2119] is a signaling protocol for secure communication, mobility, and multihoming by introducing a
cryptographic namespace. This document specifies an extension for HIP that enables middleboxes to unambiguously verify the identities of hosts that communicate across them. This extension enables middleboxes to verify the liveness and freshness of a HIP association and, thus, enables reliable and secure access control in middleboxes.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Notation

[x] indicates that x is optional.

{x} indicates that x is under signature.

Initiator is the host which initiates a HIP association (cf. HIP base protocol).

Responder is the host which responds to the INITIATOR (cf. HIP base protocol).

--> signifies "Initiator to Responder" communication.

<-- signifies "Responder to Initiator" communication.
Table of Contents

1. Introduction ................................................. 4
   1.1. Authentication and Replay Attacks ....................... 5
2. Protocol Overview ........................................ 6
   2.1. Signed Middlebox Nonces ................................. 6
   2.2. Identity Verification by Middleboxes .................... 8
   2.3. Failure Signaling ...................................... 13
   2.4. Fragmentation .......................................... 13
   2.5. HIP Parameters ........................................ 13
3. Security Services for the HIP Control Channel ............. 16
   3.1. Adversary model and Security Services ................... 16
4. Security Services for the HIP Payload Channel ............ 17
   4.1. Access Control ........................................ 18
   4.2. Resource allocation .................................... 19
5. Security Considerations .................................... 19
6. IANA Considerations ........................................ 20
7. Acknowledgments ............................................ 20
8. Normative References ...................................... 20
Authors’ Addresses ............................................. 21
Intellectual Property and Copyright Statements ................ 22
1. Introduction

The Host Identity Protocol (HIP) introduces a new cryptographic namespace, based on public keys, in order to secure Internet communication. This namespace allows hosts to authenticate their peers. HIP was designed to be middlebox-friendly and allows middleboxes to inspect HIP control traffic. Examples of such middleboxes are firewalls and Network Address Translators (NATs).

In this context, one can distinguish HIP-aware middleboxes, which were designed to process HIP packets, and other middleboxes, which are not aware of the Host Identity Protocol. This document addresses only HIP-aware middleboxes while the behavior of HIP in combination with non-HIP-aware middleboxes is specified in [I-D.ietf-hip-nat-traversal]. Moreover, the scope of this document is restricted to middleboxes that use HIP in order to enforce access control and, thus, need to authenticate the communicating peers that send traffic over the middlebox. The class of middleboxes this document focuses on does not require the end-host to establish an explicit registration with the middlebox. HIP behavior for interacting and registering to such middleboxes is specified in [I-D.ietf-hip-registration]. Thus, we focus on middleboxes that build their state based on packets they forward.

An example of such a middlebox is a firewall that only allows traffic from certain hosts to traverse. We assume that access control is performed based on Host Identities (HIs). Such an authenticating middlebox needs to observe the HIP Base EXchange (BEX) or a HIP mobility update [I-D.ietf-hip-mm] and check the Host Identifiers (HIs) in the packets.

Along the lines of [I-D.irtf-hiprg-nat], an authentication solution for middleboxes must have some vital properties. For one, the middlebox must be able to unambiguously identify one or both of the communicating peers. Additionally, the solution must not allow for new attacks against the middlebox. This document specifies a HIP extension that allows middleboxes to participate in the HIP handshake and the HIP update process in order to enable these middleboxes to reliably verify the identities of the communicating peers. To this end, this HIP extension defines how middleboxes can interact with end-hosts in order to verify their identities.

Verifying public-key (PK) signatures is costly in terms of CPU cycles. Thus, in addition to authentication capabilities, it is also necessary to provide middleboxes with a way of defending against resource-exhaustion attacks that target PK signature verification. This document defines how middleboxes can utilize the HIP puzzle mechanism defined in [I-D.ietf-hip-base] to slow down resource-
exhaustion attacks.

The presented authentication extension only targets the HIP control channel. Additional security considerations and possible security services for the HIP payload channel are discussed in Section 4.

1.1. Authentication and Replay Attacks

Middleboxes may need to verify the HIs in the HIP base exchange messages to perform access control based on Host Identities. However, passive verification of HIs in the messages is not sufficient to verify the identity of an end-host because of replay attacks. The basic HIP protocol as specified in [I-D.ietf-hip-base] does not provide adequate protection against these attacks.

To illustrate the need for additional security requirements with HIP-aware middleboxes, we briefly outline a possible replay attack targeted at middleboxes. Assume that a middlebox M checks HIP HIs in order to restrict traffic passing through the box. Further assume that the legitimate owner of Host Identity Tag (HIT) X establishes a HIP association with the legitimate owner of HIT Y at some point in time and an attacker A overhears the base exchange and records it. Note that it is not required that the middlebox M is on the communication path between the peers at that time.

At some later point in time, Attacker A collaborates with another attacker B. They replay the very same BEX with the middlebox M on the communication path. The middlebox has no way to distinguish legitimate hosts X and Y from the attackers A and B as it can only overhear the BEX passively and does not participate in the authentication process. As the attackers overheard the SPI numbers, they can bypass the middlebox with "fake" ESP packets with valid ESP numbers. Since the middleboxes do not know the integrity and encryption keys for ESP, they cannot distinguish valid ESP packets from fake ones. Hence, collaborating attackers can use any recorded BEX to falsely authenticate to the middlebox and thus impersonate any host. This is problematic in cases in which the middlebox needs to know the identity of the peers that communicate across it. Examples for such cases are access control, logging of activities, and accounting for traffic volume or connection duration.

This attack scenario is not addressed by the current HIP specifications. Therefore, this document specifies a HIP extension that allows middleboxes to defend against it.
2. Protocol Overview

This section gives an overview of the interaction between hosts and authenticating middleboxes. This document describes a framework that middleboxes can use to implement authentication of end-hosts and leaves its further use to other documents and to middlebox implementors.

2.1. Signed Middlebox Nonces

The described attack scenario shows the necessity for unambiguous end-host identity verification by middleboxes. Relying on nonces generated by the end-hosts is not possible because middleboxes cannot verify the freshness of these nonces. Introducing time-stamps restricts the attack to a certain time frame but requires global time synchronization.

The following sections specify how HIP hosts can prove their identity by performing a challenge-response protocol between the middlebox and the end-hosts. As the challenge, the middlebox adds information (e.g. nonces) to HIP control packets which the end-hosts sign with public-key (PK) signatures and echo back.

The challenge-response mechanism is similar to the ECHO_REQUEST/ECHO_RESPONSE mechanism used by HIP end-hosts to authenticate their peers. It assumes that the end-hosts exchange at least two HIP packets with each other. The middlebox adds an ECHO_REQUEST_M parameter to the first HIP control packet that contains a nonce. The peer host receives the first packet and processes it normally. However, the peer will also include an ECHO_RESPONSE_M in the second message which contains the nonce from the ECHO_REQUEST_M. Before sending the second message, the peer also signs it to prove that it is in the possession of the private key that corresponds its HI.

The middlebox can either verify the identity of the initiator, the responder, or both peers, depending on the purpose of the middlebox. The choice which authentication is required left to middlebox implementers.

2.1.1. ECHO_REQUEST_M

Middleboxes MAY add ECHO_REQUEST_M parameters to the the R1, I2, and to any UPDATE packet. This parameter contains an opaque data block of variable size which the middlebox uses to carry arbitrary data. The HIP packets allowed to carry middlebox nonces may contain multiple ECHO_REQUEST_M parameters. As all middleboxes on the path may add ECHO_REQUEST_M parameters, the length of the data field of each parameter SHOULD not exceed a maximum of 32 bytes. The total
length of the packets SHOULD not exceed 1280 bytes to avoid IPv6 fragmentation.

The middleboxes add the ECHO_REQUEST_M parameter to the unprotected part of a HIP message. Thus it does not corrupt any HMAC or public-key signatures. However, the middlebox MUST recompute the IP- and HIP header checksums as defined in [I-D.ietf-hip-base] and the UDP headers of UDP encapsulated HIP packets as defined in [I-D.ietf-hip-nat-traversal].

An end-host that receives a HIP control packet containing one or multiple ECHO_REQUEST_M parameters must copy the contents of each parameter without modification to an ECHO_RESPONSE_M parameter. This end-host MUST send this parameter within the signed part of its reply. Note that middleboxes MAY also rewrite the ECHO_REQUEST_UNSIGNED parameter as specified in [I-D.ietf-hip-base] when the receiver of the parameter does not have to sign the contents of the ECHO_REQUEST.

Middleboxes can delay state creation by utilizing the ECHO_RESPONSE_M and ECHO_REQUEST_M parameter by hiding encrypted or otherwise protected information about previous authentication steps in the opaque blob.

2.1.2. ECHO_RESPONSE_M

When a middlebox injects an opaque blob of data via an ECHO_REQUEST_M parameter, it expects to receive the same data without modification as part of an ECHO_RESPONSE_M parameter in a subsequent packet. The opaque data MUST be copied as it is from the corresponding ECHO_REQUEST_M parameter. In the case of multiple ECHO_REQUEST_M parameters, their order MUST be preserved by the corresponding ECHO_RESPONSE_M parameters.

The ECHO_REQUEST_M and ECHO_RESPONSE_M parameters MAY be used for any purpose, in particular when a middlebox needs to carry state information in a HIP packet and receive it in a subsequent response packet. The ECHO_RESPONSE_M MUST be covered by the HIP_SIGNATURE.

The ECHO_RESPONSE_M parameter is non-critical. Depending on its local policy, a middlebox can react differently on a missing ECHO_RESPONSE_M parameter. Possible actions range from degraded or restricted service such as bandwidth limitation up to refusing connections and reporting access violations.
2.1.3. Middlebox Puzzles

As PK operations are costly in terms of CPU cycles, a middlebox needs to defend itself against resource-exhaustion attacks. The HIP base protocol [I-D.ietf-hip-base] specifies a puzzle mechanism to protect the Responder from I2 floods that require numerous public-key operations. However, middleboxes cannot utilize this mechanism as there is no defense against a collaborative replay attack, which involves a malicious Initiator and a malicious Responder. This section specifies how middleboxes can utilize the puzzle mechanism to add their own puzzles to R1, I2, and any UPDATE packets. This allows middleboxes to shelter against Denial of Service (DoS) attacks on PK verification.

To defend against such attacks, a middlebox adds a puzzle in a PUZZLE_M parameter to I2, R2 and UPDATE packets. The destination end-host of the HIP control packet must solve it.

As a puzzle increases the delay and computational cost for establishing or updating a HIP association, a middlebox SHOULD only add puzzles to packets when it is under attack. Moreover, middleboxes SHOULD distinguish attack directions. If the majority of the CPU load is caused by verifying HIP control messages that arrive from a certain interface, middleboxes MAY add puzzles to HIP control packets that leave the interface. The middlebox chooses the difficulty of the puzzle according to its load and local policies.

Middleboxes MAY decide to use just the PUZZLE_M parameter instead of using PUZZLE_M in combination with ECHO_REQUEST_M because the PUZZLE_M parameter also contains an opaque data field that guarantees the freshness of the signature. However, the opaque data field in the PUZZLE_M and the corresponding SOLUTION_M parameter is restricted to 6 bytes which may not be sufficient for all purposes.

2.2. Identity Verification by Middleboxes

This section describes how middleboxes can influence the BEX and the HIP update process in order to verify the identity of the HIP end-hosts.

2.2.1. Identity Verification During BEX

Middleboxes MAY add ECHO_REQUEST_M and PUZZLE_M parameters to R1 and I2 packets in order to verify the identities of the participating parties. Middleboxes can choose either to authenticate the Initiator, the Responder, or both. Middleboxes MUST NOT add ECHO_REQUEST_M or PUZZLE_M parameters to I1 messages because this would expose the Responder to DoS attacks. Thus, middleboxes MUST
let unauthenticated and minimal I1 packets traverse. Minimal means that the packet MUST NOT contain more than the minimal set of parameters specified by HIP standards or internet drafts. In particular, the I1 packet MUST NOT contain any attached payload. Figure 1 illustrates the authentication process during the BEX.

Middlebox authentication of a HIP base exchange.

Main path:

<table>
<thead>
<tr>
<th>Initiator</th>
<th>Middlebox</th>
<th>Responder</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1</td>
<td>I1</td>
<td>R1</td>
</tr>
<tr>
<td>R1, + EQ1, [PM1]</td>
<td>Add EQ1, PM1</td>
<td>R1</td>
</tr>
<tr>
<td>I2, {ER1, [SM1]}</td>
<td>Verify SM1, EQ1</td>
<td>I2, {ER1, [SM1]} + EQ2, [PM2]</td>
</tr>
<tr>
<td>R2, {ER2, [SM2]}</td>
<td>Verify SM2, ER2</td>
<td>R2, {ER2, [SM2]}</td>
</tr>
</tbody>
</table>

EQ: Middlebox Echo reQuest
ER: Middlebox Echo Response
PM: Puzzle of the Middlebox
SM: Solution of Middlebox puzzle

Figure 2

2.2.2. Identity Verification During Mobility Updates

HIP rekeying, mobility and multihoming UPDATE mechanisms for non-NATted environments are described in [I-D.ietf-hip-mm]. This section describes how middleboxes process UPDATE messages in non-NATted environments and leave NATted environments for future revisions of the draft.

The middleboxes can apply middlebox nonces and puzzles to mobility related HIP control messages in the case where both end-hosts are single-homed. The middlebox nonces and puzzles can be applied both ways as the UPDATE process consists of three packets (U1, U2, U3) which all traverse through the same middlebox as shown in Figure 3.
In cases in which fewer packets are used for updating an association the following rule applies.

RESPONSE RULE:

A HIP host, receiving an ECHO_REQUEST_M MUST reply an ECHO_RESPONSE_M in its next UPDATE packet. If no further UPDATE packets are necessary to complete the update procedure, an additional UPDATE packet containing the ECHO_RESPONSE_M MUST be sent.

Middlebox authentication of a HIP mobility update over different paths.

<table>
<thead>
<tr>
<th>Initiator</th>
<th>Middlebox 1</th>
<th>Responder</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td></td>
<td>U1 + EQ1, [PM1]</td>
</tr>
<tr>
<td>U2, {ER1, [SM1]} + EQ2, [PM2]</td>
<td>OK</td>
<td>U2, {ER1, [SM1]}</td>
</tr>
<tr>
<td>U3, {ER2, [SM2]}</td>
<td>OK</td>
<td>U3, {ER2, [SM2]}</td>
</tr>
</tbody>
</table>

EQ: Middlebox Echo reQuest
ER: Middlebox Echo Response
PM: Puzzle of the Middlebox
SM: Solution of Middlebox puzzle

Figure 3

Middlebox 1 can verify the identity of the Responder by checking its PK signature and the presence of the ECHO_RESPONSE_M in the U2 packet. If necessary, the middlebox MAY add an ECHO_REQUEST_M for the Initiator of the update. The middlebox can verify the Initiator’s identity by verifying its signature and the ECHO_RESPONSE_M in the U3 packet.

A middlebox that is not located on the path between preferred locators of the HIP end-hosts does not receive the U1 message. Therefore, it will not recognize any ER1 or SM1 in the second UPDATE packet. Thus, if a middlebox encounters non-matching or missing ECHO_RESPONSE_M parameters, the middlebox SHOULD ignore these.
When receiving an UPDATE message with an ECHO_REQUEST_M, a HIP host SHOULD send an UPDATE message containing the corresponding ECHO_RESPONSE_M covered by a HIP_SIGNATURE parameter. Otherwise the middlebox may refuse to make the communication path available to the HIP host.

2.2.3. Identity Verification for Multihomed Mobility Updates

Multihomed hosts may use multiple communication paths during an HIP mobility update. Depending on whether the middlebox is located on the communication path between the preferred locators or not, the middlebox forwards different packets and, thus, needs to interact differently with the updates. Figure 4 I) and II) illustrates an update with Middlebox 1 on the path between the Initiator’s and the Responder’s preferred locators and with Middlebox 2 on an alternative path. Middlebox 2 is not located on the path between the preferred locators of the HIP end-hosts does not receive the U1 message. Therefore, it will not recognize any ER1 or SM1 in the second UPDATE packet. Thus, if a middlebox encounters non-matching or missing ECHO_RESPONSE_M parameters, the middlebox SHOULD ignore these.

Complying to the RESPONSE RULE stated in Section Section 2.2.2, the RESPONDER generates an additional fourth update packet on receiving the ECHO_REQUEST_M. The update process for a middlebox on the preferred communication path (Middlebox 1) and a middlebox off the preferred communication path (Middlebox 2) is depicted in Figure 4.
Middlebox authentication of a HIP mobility update over different paths.

I) Main path:

Initiator | Middlebox 1 | Responder
---|---|---
U1 | | U1 + EQ1, [PM1]
U2, {ER1, [SM1]} + EQ2, [PM2] | OK | U2, {ER1, [SM1]}
U3, {ER2, [SM2]} | OK | U3, {ER2, [SM2]}

II) Alternative path:

Initiator | Middlebox 2 | Responder
---|---|---
U1 (bypasses Middlebox 2) | | 
U2, {ER1, [SM1]} + EQ3, [PM3] | wrong | U2, {ER1, [SM1]}
U3’, {ER3, [SM3]} | OK | U3’, {ER3, [SM3]} + EQ4, PM4
U4, {ER4, [SM4]} | OK | U4, {ER1, [SM1]}

EQ: Middlebox Echo reQuest
ER: Middlebox Echo Response
PM: Puzzle of the Middlebox
SM: Solution of Middlebox puzzle

Figure 4

2.2.4. Identity Signaling During Updates

As middleboxes need to be able to rapidly verify and forward HIP packets, they need to be supplied with all information necessary to do so. If end-host hand over communication to a new communication path, middleboxes need to be able to learn the Host Identifiers (HIs)
from the UPDATE packets. Therefore, HIP end-hosts MUST include the
HOST_ID parameter in all UPDATE packets that use combinations of
locators that have not been used before. Additionally, UPDATE
packets that contain ECHO_REQUEST or ECHO_RESPONSE parameters MUST
contain the HOST_ID parameter. Moreover, all packets that contain an
ECHO_RESPONSE_M parameter MUST contain the HOST_ID parameter.

2.2.5. Closing of Connections

At the time being, identity verification during the closing of a HIP
association is not supported. Hence, the middlebox MUST preserve the
state until it expires according to local policies. An appropriate
mechanism for middleboxes to verify CLOSE messages by middleboxes
will be provided in future versions of this document.

2.3. Failure Signaling

Middleboxes SHOULD inform the sender of a BEX or update message if it
does not satisfy the requirements of the middlebox. Reasons for non-
satisfactory packets are missing HOST_ID, ECHO_RESPONSE_M, and
SOLUTION_M parameters. Options for expressing such shortcomings are
ICMP packets if no HIP association is established and HIP_NOTIFY
packets in case of an already established HIP association. Defining
this signaling mechanism is future work.

2.4. Fragmentation

Analogously to the specification in [I-D.ietf-hip-base], HIP aware
middleboxes SHOULD support IP-level fragmentation and reassembly for
IPv6 and MUST support IP-level fragmentation and reassembly for IPv4.
However, when adding ECHO_REQUEST_M and PUZZLE_M parameters, a
middlebox SHOULD keep the total packet size below 1280 bytes to avoid
packet fragmentation in IPv6.

2.5. HIP Parameters

This HIP extension specifies four new HIP parameters that allow
middleboxes to authenticate HIP end-hosts and to protect against DoS
attacks.

2.5.1. ECHO_REQUEST_M

A middlebox MAY apply ECHO_REQUEST_M parameter to R1, I2, and UPDATE
packets. The structure of the ECHO_REQUEST_M parameter is depicted
in the following figure.
Type 65332
Length Variable
Opaque data Opaque data, should be interpreted only by the middlebox that adds ECHO_REQUEST_M and receives the corresponding ECHO_RESPONSE_M.

2.5.2. ECHO_RESPONSE_M

The ECHO_RESPONSE_M is the reply to the ECHO_REQUEST_M parameter. The receiver of an ECHO_REQUEST_M parameter SHOULD reply with n ECHO_RESPONSE_M. Otherwise, the middlebox that added the parameter MAY decide to degrade or deny its service. The contents of the ECHO_REQUEST_M parameter must be copied to the ECHO_RESPONSE_M parameter without any modification. The ECHO_RESPONSE_M parameter is non-critical and covered by the SIGNATURE. The structure of the ECHO_RESPONSE_M parameter is depicted below:

Type 962
Length Variable
Opaque data Opaque data, should be interpreted only by the middlebox that adds ECHO_REQUEST_M and receives the corresponding ECHO_RESPONSE_M.

2.5.3. PUZZLE_M

A middlebox MAY add a PUZZLE_M parameter to R1, I2, and UPDATE packets. A HIP packet may contain multiple PUZZLE_M parameters as multiple middleboxes may be located on a communication path. These
puzzles serve as defense against DoS attacks. Hosts that receive a PUZZLE_M parameter SHOULD reply with a SOLUTION_M parameter in the subsequent I2, R2, or UPDATE packet. With the exception of an extended opaque field, the syntax and semantics of the puzzle are defined in [I-D.ietf-hip-base]. The extended opaque data field helps middleboxes to recognize their puzzles and solutions, respectively, when a packet contains more than one puzzle.

A middlebox MUST preserve the order of PUZZLE_M parameters in a packet and attach its own PUZZLE_M parameter after all other PUZZLE_M parameters. Preserving the order of PUZZLE_M parameters may speed up the middlebox recognition of the puzzles.

2.5.4. SOLUTION_M

The SOLUTION_M parameter contains the solution for the corresponding PUZZLE_M parameter. End-hosts that receive a PUZZLE_M parameter SHOULD solve the puzzle according to the specification in [I-D.ietf-hip-base] and send the resulting solution in the SOLUTION_M parameter. Exclusion of a solution MAY result in degraded or denied service by the middlebox that added the PUZZLE_M parameter. The format and meaning of the fields in the SOLUTION_M parameter resemble the specifications of the SOLUTION parameter in [I-D.ietf-hip-base]. The reader is advised to refer to that document for further details.

The extended opaque data field helps middleboxes to recognize their
puzzles and the resulting solutions, respectively, when a packet contains multiple puzzles.

The relative order of SOLUTION_M parameters in a HIP control packet MUST match the order of the PUZZLE_M parameters in the previously received packet. Preserving the order of PUZZLE_M for the corresponding SOLUTION_M parameters may help middleboxes to recognize the puzzles and solutions relevant to them.

```
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
     +---------------------------------------------+
     |             Type              |             Length            |
     +---------------------------------------------+
     | K, 1 byte     |    Reserved   |        Opaque, 6 bytes        |
     +---------------------------------------------+
     | Random #I, 8 bytes                                           |
     +---------------------------------------------+
     | Puzzle solution #J, 8 bytes                                   |
```

<table>
<thead>
<tr>
<th>Type</th>
<th>322</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>20</td>
</tr>
<tr>
<td>K</td>
<td>K is the number of verified bits</td>
</tr>
<tr>
<td>Reserved</td>
<td>Zero when sent, ignored when received</td>
</tr>
<tr>
<td>Opaque</td>
<td>Copied unmodified from the received PUZZLE parameter</td>
</tr>
<tr>
<td>Random #I</td>
<td>Random number</td>
</tr>
<tr>
<td>Puzzle solution</td>
<td>Random number</td>
</tr>
</tbody>
</table>

3. Security Services for the HIP Control Channel

In this section, we define the attacker model that the security analysis in the later sections will be based on.

3.1. Adversary model and Security Services

For discussing the security properties of the proposed HIP extension we first define an attacker model. We assume a Dolev-Yao threat
model in which an adversary can eavesdrop on all traffic regardless of its source and destination. The adversary can inject arbitrary packets with any source and destination addresses. Consequently, an adversary can also replay previously eavesdropped messages. However, the adversary cannot subvert the cryptographic ciphers and hash function, nor can it take over one of the communicating nodes.

Even in the face of this strong attacker, the proposed HIP extension enables middleboxes to verify the identity of the communicating HIP peers. It ensures that both peers are involved in the communication and that the HIP BEX or update packets are fresh, i.e. not replayed. It enables the middlebox to verify the source and destination (in terms of HIs) of the HIP association and the integrity of RSA and DSA signed HIP packets.

4. Security Services for the HIP Payload Channel

The presented extension for HIP authentication by middleboxes only covers the HIP control channel, i.e., the HIP control messages. Depending on the binding between the HIP control and payload channel, certain security properties for the payload channel can be derived from the strong cryptographic authentication of the end-hosts. Assuming that there is a secure binding between packets belonging to a payload stream and the control stream, the same security properties as in Section 3 apply to the payload stream.

ESP [I-D.ietf-hip-esp] is currently the default payload encapsulation format for HIP. A limitation of ESP is that does not provide a secure binding between the HIP control channel and the ESP traffic on a per-packet basis, the achievable level of security for the payload channel is lower.

This section discusses security properties of an ESP payload channel bound to a HIP control channel. Depending on the assumed adversary model, certain security services are possible. We briefly describe two application scenarios and how they benefit from the resulting security services. For the payload channel, HIP in combination with the middlebox authentication scheme offers the following security services:

Attribute binding: Middleboxes can extract certain payload channel attributes (e.g. locators and SPIs) from the control channel. These attributes can be used to enforce certain restrictions on the payload channel, e.g., to exhibit the same attributes as the control channel. The attributes can either be stated explicitly in the HIP control packets or can be derived from the IP or UDP packets carrying the HIP control messages.
Host involvement: Middleboxes can verify whether a certain host was involved in the establishment of a HIP association and thus, in the establishment of the payload channel.

Based on these security services we construct two use cases that illustrate the use of HIP authentication by middleboxes: access control and resource allocation as described in the following sections.

4.1. Access Control

Middleboxes can manage resources based on HIs. As an example, let us assume that a middlebox only forwards HIP payload packets after a successful HIP BEX or HIP update. The middlebox uses the parameters in the control channel (specifically IP addresses and SPIs) to filter the payload traffic. The middlebox only forwards traffic from and to specific authenticated hosts and drops other traffic.

The feasibility of subverting the function of the middlebox depends on the assumed adversary model.

4.1.1. Adversary model and Security Services

If we assume a Dolev-Yao threat model, attribute binding is not helpful to aid packet filtering for access control. An attacker can send packets from any IP address and can read packets destined to any IP address. Without per packet verification by the middlebox, such an attacker can inject arbitrary forged packets into the HIP payload channel and make them traverse the middlebox. The attacker can also read the packets from the HIP payload channel, and hence, communicate across the middlebox. However, the injected packets are disclosed by inconsistencies in the ESP sequence numbers, which makes the attack visible to the middlebox as well as the HIP end hosts. Moreover, attackers can only inject packets into an already established HIP payload channel. Opening a new payload channel and replaying a closing of the channel are not possible.

An attacker that is not able to send IP packets from an arbitrary source address and receive IP packets addressed to any destination, cannot use the ESP channel to send fake ESP packets when the middleboxes bind HIs and SPI numbers to addresses. By fixing the set of source and destination IP addresses, the opportunity to successfully inject packets into the payload channel is limited to hosts that can send packets from the same source address as the legitimate HIP hosts. Moreover, an attacker can only receive injected packets if it is on the communication path towards the legitimate HIP peer. Attackers cannot open new HIP payload channels and thus have no influence on the bound payload stream parameters.
Finally, attackers cannot close HIP associations of legitimate peers.

4.2. Resource allocation

When using HIs to limit the resources (e.g. bandwidth) allocated for a certain host, the HIs can be used to authenticate the hosts in a similar fashion to the access control illustrated above. Regarding authentication, both use cases share the same strengths and weaknesses. However, the implications for the targeted scenarios differ. Therefore, we restrict the following discussion to these differences.

4.2.1. Adversary Model and Security Services

When assuming a Dolev-Yao threat model, an attacker is able to use resources allocated for the payload channel of another host by injecting packets into this channel. However, the attacker cannot open a new payload channel with another host nor can it close an existing one. When binding the IP addresses of the HIP payload channel to the IP addresses used in the HIP control channel and assuming an attacker that cannot receive IP packets addressed to the IP address of an authenticated host, the attacker cannot utilize the resources allocated to authenticated host. However, the attacker can still inject packets and waste resources, yet without having any benefit other than causing disturbance to the other host. Specifically, it cannot increase the share of resources allocated to itself. Hence, this measure takes incentive from selfish users that try to benefit by mounting a DoS attack. Defense against purely malicious attackers that aim at creating disturbance without immediate benefit is difficult to achieve.

5. Security Considerations

This HIP extension specifies how HIP-aware middleboxes interact with the handshake and mobility-signaling of the Host Identity Protocol. Its scope is restricted to the authentication of end-hosts and does not include the issue of authenticating ESP traffic at the middlebox.

Providing middleboxes with a way of adding puzzles to the HIP control packets may cause both HIP peers, including the Responder, to spend CPU time on solving these puzzles. Thus, it is advised that HIP implementations for servers employ mechanisms to prevent middlebox puzzles from being used as DoS attacks. Under high CPU load, servers can rate limit or assign lower priority to packets containing middlebox puzzles.

If multiple middleboxes add ECHO_REQUEST_M parameters to a HIP
control packet, the remaining space in the packet might not be sufficient for further parameters to be added. Moreover, as the ECHO_REQUEST_M must be echoed within an ECHO_RESPONSE_M, the space in the subsequent packet may not be sufficient to include all ECHO_RESPONSE_M parameters. Thus, middleboxes SHOULD keep the size of the nonces small.

6. IANA Considerations

This document specifies four new HIP parameter types. The preliminary parameter type numbers are 322, 962, 65332, and 65334.

7. Acknowledgments

Thanks to Shaohui Li, and Janne Lindqvist for the fruitful discussions on this topic. Many thanks to Stefan Goetz, Ari Keranen, Samu Varjonen, Rene Hummen, and Kate Harrison for commenting and helping to improve the quality of this document.

8. Normative References

[I-D.ietf-hip-base]

[I-D.ietf-hip-esp]

[I-D.ietf-hip-mm]
Henderson, T., "End-Host Mobility and Multihoming with the Host Identity Protocol", draft-ietf-hip-mm-05 (work in progress), March 2007.

[I-D.ietf-hip-nat-traversal]

[I-D.ietf-hip-registration]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

Authors' Addresses

Tobias Heer (editor)
Distributed Systems Group, RWTH Aachen University
Ahornstrasse 55
Aachen 52062
Germany

Phone: +49 241 80 214 36
Email: heer@cs.rwth-aachen.de
URI: http://ds.cs.rwth-aachen.de/members/heer

Klaus Wehrle
Distributed Systems Group, RWTH Aachen University
Ahornstrasse 55
Aachen 52062
Germany

Phone: +49 241 80 214 30
Email: wehrle@cs.rwth-aachen.de
URI: http://ds.cs.rwth-aachen.de/members/klaus

Miika Komu
Helsinki Institute for Information Technology
Metsanneidonkuja 4
Espoo
Finland

Phone: +358503841531
Fax: +35896949768
Email: miika@iki.fi
URI: http://www.hiit.fi/