Host Identity Protocol
Internet-Draft
Intended status: Experimental
Expires: April 30, 2012

T. Heer, Ed.
R. Hummen
K. Wehrle
RWTH Aachen University,
Communication and Distributed
Systems Group
M. Komu
Aalto University
October 28, 2011

End-Host Authentication for HIP Middleboxes
draft-heer-hip-middle-auth-04

Abstract

The Host Identity Protocol [RFC5201] is a signaling protocol for secure communication, mobility, and multihoming that introduces 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 allows middleboxes to verify the liveness and freshness of a HIP association and, thus, to 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 [RFC2119].

Notation

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on April 30, 2012.

Copyright Notice

Table of Contents

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 securely address and authenticate their peers. HIP was designed to be middlebox-friendly and to allow 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 are designed to process HIP packets, and other middleboxes, which are unaware of HIP. This document addresses only HIP-aware middleboxes while the behavior of HIP in combination with HIP-unaware middleboxes is specified in [RFC5770]. Moreover, the scope of this document is restricted to middleboxes that use HIP in order to provide Authentication, Authorization, and Accounting (AAA)-related services 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 explicitly register to the middlebox. HIP behavior for interacting and registering to such middleboxes is specified in [RFC5203]. Thus, we focus on middleboxes that build their state based on packets they forward (path-coupled signaling).

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 [RFC5206] and check the Host Identifiers (HIs) in the packets.

Along the lines of [RFC5207], 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 allow 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 [RFC5201] 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 ensure the identity of an end-host because of a possible replay attack against which the basic HIP protocol as specified in [RFC5201] does not provide adequate protection.

To illustrate the need for additional security measures for HIP-aware middleboxes, we briefly outline the replay attack: 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.

Assume that a middlebox M checks HIP HIs in order to restrict traffic passing through the box. At some later point in time, Attacker A collaborates with another attacker B. They replay the very same BEX packets to the middlebox M on the communication path. Note that it is not required that the middlebox M was on the communication path between X and Y when the BEX was recorded.

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 it cannot can distinguish the replayed BEX from a the genuine handshake. As the attackers overheard the SPI numbers, they can traverse the middlebox with "fake" ESP packets with valid SPI numbers, and hence, send data across M without proper authentication. Since the middleboxes do not know the integrity and encryption keys for ESP, they cannot distinguish valid ESP packets from forged ones. Hence, collaborating attackers can use any replayed 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 AAA-related services, such as 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 this attack.

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. However, this authentication cannot be purely end-to end: a) Relying on nonces generated by the end-hosts is not possible because middleboxes cannot verify the freshness of these nonces. b) Introducing time-stamps restricts the attack to a certain time frame but requires global time synchronization and therefore should be avoided.

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 a challenge, the middlebox adds information (e.g. self-generated 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 employed already by HIP end-hosts (see [RFC5201]). It assumes that the end-hosts exchange at least two HIP packets with each other. The middlebox adds a CHALLENGE_REQUEST parameter to the first HIP control packet. Similar to the ECHO_REQUEST parameter in the original HIP protocol, this parameter contains an opaque data field that must be echoed by its receiver. The receiver echoes the opaque data field in a CHALLENGE_RESPONSE parameter. The CHALLENGE_RESPONSE parameter must be covered by the packet signature, thereby proving that the receiver is in possession of the private key that corresponds to the 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 of which authentication is required left to middlebox implementers.

2.1.1. CHALLENGE_REQUEST

Middleboxes MAY add CHALLENGE_REQUEST parameters to the R1 and I2 packets and to any UPDATE packet. This parameter contains an opaque data block of variable size, which the middlebox uses to carry arbitrary data (e.g., a nonce). The HIP packets that carry middlebox challenges may contain multiple CHALLENGE_REQUEST parameters, since

all middleboxes on the path may add these parameters. A middlebox MUST append its own CHALLENGE_REQUEST parameter behind already existing CHALLENGE_REQUEST parameters in the HIP packet. In order to avoid packet fragmentation, the MBs should restrict the size of the variable data field in the CHALLENGE_REQUEST parameter. The total length of the packets SHOULD not exceed 1280 bytes to avoid IPv6 fragmentation [RFC2460].

The middleboxes add the CHALLENGE_REQUEST parameter to the unprotected part of a HIP message. Thus, it does not corrupt any HMAC or public-key signatures that protect the HIP packet. However, the middlebox MUST recompute the IP and HIP header checksums as defined in [RFC5201] and the UDP headers of UDP encapsulated HIP packets as defined in [RFC5770].

A HIP end-host that receives a HIP control packet containing one or more CHALLENGE_REQUEST parameters must copy the contents of each parameter without modification to a single CHALLENGE_RESPONSE parameter. This end-host MUST send the CHALLENGE_RESPONSE parameter within the signed part of its reply. Note that middleboxes MAY also add ECHO_REQUEST_UNSIGNED parameters as specified in [RFC5201] if the receiver of the parameter is not required to sign the contents of the ECHO_REQUEST.

Middleboxes can delay state creation by utilizing the CHALLENGE_REQUEST and CHALLENGE_RESPONSE parameters by hiding encrypted or otherwise protected information about previous authentication steps in the opaque data field.

2.1.2. CHALLENGE_RESPONSE

When a middlebox injects an opaque blob of data with a CHALLENGE_REQUEST parameter, it expects to receive the same data without modification as part of a CHALLENGE_RESPONSE parameter in a subsequent packet. Hence, the opaque data MUST be copied as it is from the corresponding CHALLENGE_REQUEST parameter. In the case of multiple CHALLENGE_REQUEST parameters, their order MUST be preserved within the corresponding CHALLENGE_RESPONSE parameter.

The CHALLENGE_REQUEST and CHALLENGE_RESPONSE parameters MAY be used for any purpose, in particular when a middlebox has to carry state information in a HIP packet to receive it in the next response packet. The CHALLENGE_RESPONSE MUST be covered by the HIP_SIGNATURE.

The CHALLENGE_RESPONSE parameter is non-critical. Depending on its local policy, a middlebox can react differently on a missing CHALLENGE_RESPONSE parameter. Possible actions range from degraded or restricted service, such as bandwidth limitation, up to refusing

connections and reporting access violations.

When sending a HIP control packet, an end-host may face the problem that not all opaque values of the received CHALLENGE_REQUEST parameters fit into the CHALLENGE_RESPONSE parameter due to HIP control packet size restrictions. In this case, the host should send several packets. The first packet contains a CHALLENGE_RESPONSE parameter that includes the received opaque values of the CHALLENGE_REQUEST parameters starting from the last occurrence in the packet. Further packets contain the remaining values in the reverse order of the inclusion in the received packet. This way, the middleboxes closest to the sender will already have authenticated the identity of the peers and can let further control packets pass through.

2.1.3. Middlebox Puzzles

Since PK operations are costly in terms of CPU cycles, a middlebox has to defend itself against resource-exhaustion attacks when verifying signatures in HIP packets. The HIP base protocol [RFC5201] specifies a puzzle mechanism to protect the Responder from I2 floods that require numerous public-key operations. However, middleboxes cannot utilize this mechanism because they cannot verify the freshness of the puzzle solution in the BEX packets. 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.

The puzzle mechanism for middleboxes utilizes the CHALLENGE_REQUEST and CHALLENGE_RESPONSE parameters. The CHALLENGE_REQUEST parameter contains fields for setting the difficulty and the expiration date of the puzzle. In contrast to the PUZZLE parameter in the HIP base specifications, there is no dedicated puzzle seed field. Instead, the hash of the opaque data field in the CHALLENGE_REQUEST parameter serves as puzzle seed. The hash is generated by applying the SHA-1 algorithm to the opaque data field. The destination end-host of the HIP control packet MUST solve the puzzle and provide the solution in the CHALLENGE_RESPONSE parameter. The middlebox can set the puzzle difficulty by adjusting the K value in the CHALLENGE_REQUEST packet. The semantics of this field equal the semantics of the PUZZLE parameter. Setting K to 0 signifies that no puzzle solution is required.

In case of multiple CHALLENGE_RESPONSE parameters, the responder derives the puzzle seed from the concatenation of the opaque data of all CHALLENGE_REQUEST parameters in the received control packet in the reverse order of their inclusion. Furthermore, he MUST compute

the solution based on the highest difficulty value K in the received CHALLENGE_REQUEST parameters. This selection of K satisfies the security requirements of each middlebox while preventing the the receiver from computing multiple puzzle solutions. The responder MUST meet the lowest time boundaries of the received CHALLENGE_REQUEST parameters. Otherwise, there exists one on-path middlebox that will not approve the solution.

When approaching the IPv6 packet fragmentation threshold, end-hosts should split the CHALLENGE_RESPONSE parameter in case of multiple CHALLENGE_REQUEST parameters. Hence, end-hosts SHOULD compute the puzzle solution after the overall packet size of the response packet has been determined. Hence, only the opaque values of the CHALLENGE_REQUEST parameters that are included in the respective CHALLENGE_RESPONSE parameter MUST be used during the puzzle seed generation.

Since a puzzle increases the delay and computational cost for establishing or updating a HIP association, a middlebox SHOULD only increase K 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 increase K for HIP control packets that leave the interface. The middlebox chooses the difficultly of the puzzle according to its load and local policies.

2.1.4. CHALLENGE_RESPONSE Verification

When a middlebox has added a CHALLENGE_REQUEST parameter to a control packet and receives a control packet that contains a CHALLENGE_RESPONSE parameter, it first checks if its opaque data has been echoed back correctly. To this end, it traverses the Opaque values included in the CHALLENGE_RESPONSE parameter.

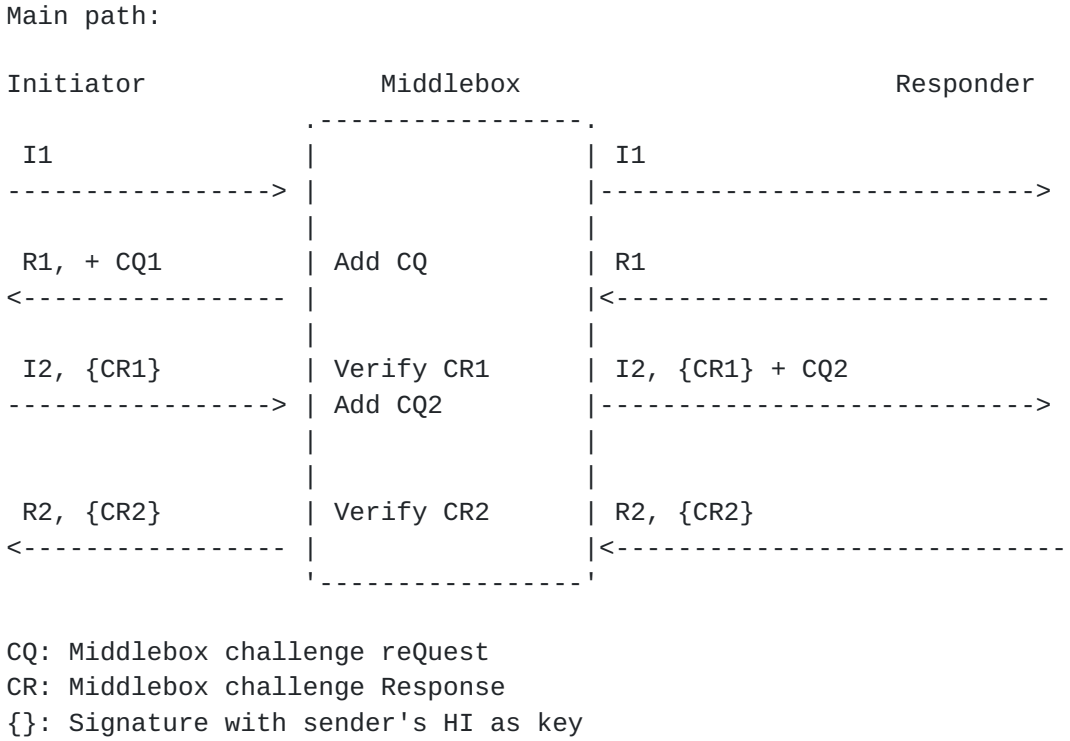
If the opaque data has been echoed back correctly by the end-host, the middlebox verifies the provided puzzle solution. It, therefore, hashes the Opaque values as contained in the CHALLENGE_RESPONSE parameter and verifies the signaled solution. In case of a successful verification, the middlebox MAY check further security mechanisms such as the PK signature and process the packet according to its function.

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 CHALLENGE_REQUEST parameters to R1 and I2 packets in order to verify the identities of the participating end-hosts. Middleboxes can choose either to authenticate the Initiator, the Responder, or both. Middleboxes MUST NOT add CHALLENGE_REQUEST 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 I1 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.

Figure 1

2.2.2. Identity Verification During Mobility Updates

HIP rekeying, mobility and multihoming UPDATE mechanisms for non-NATted environments are described in [RFC5206]. This section describes how middleboxes process UPDATE messages in non-NATted environments and leave NATted environments for future revisions of

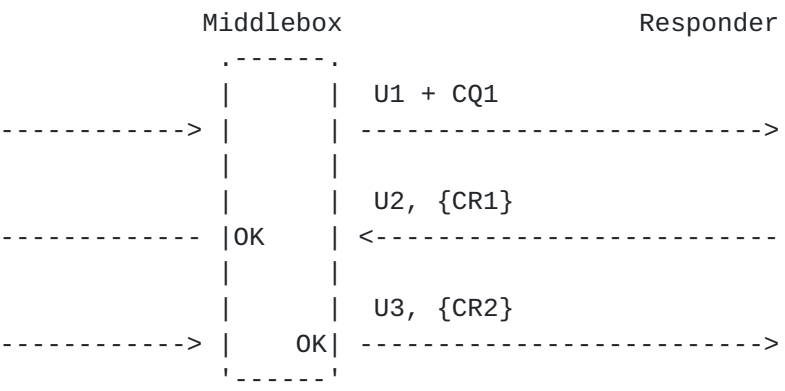
the draft.

The middleboxes can apply middlebox challenges to mobility related HIP control messages in the case where both end-hosts are single-homed. The middlebox challenges 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 2.

In cases, in which fewer packets are used for updating an association, the following rule applies.

RESPONSE RULE:

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



Middlebox authentication of a HIP mobility update over a single path.

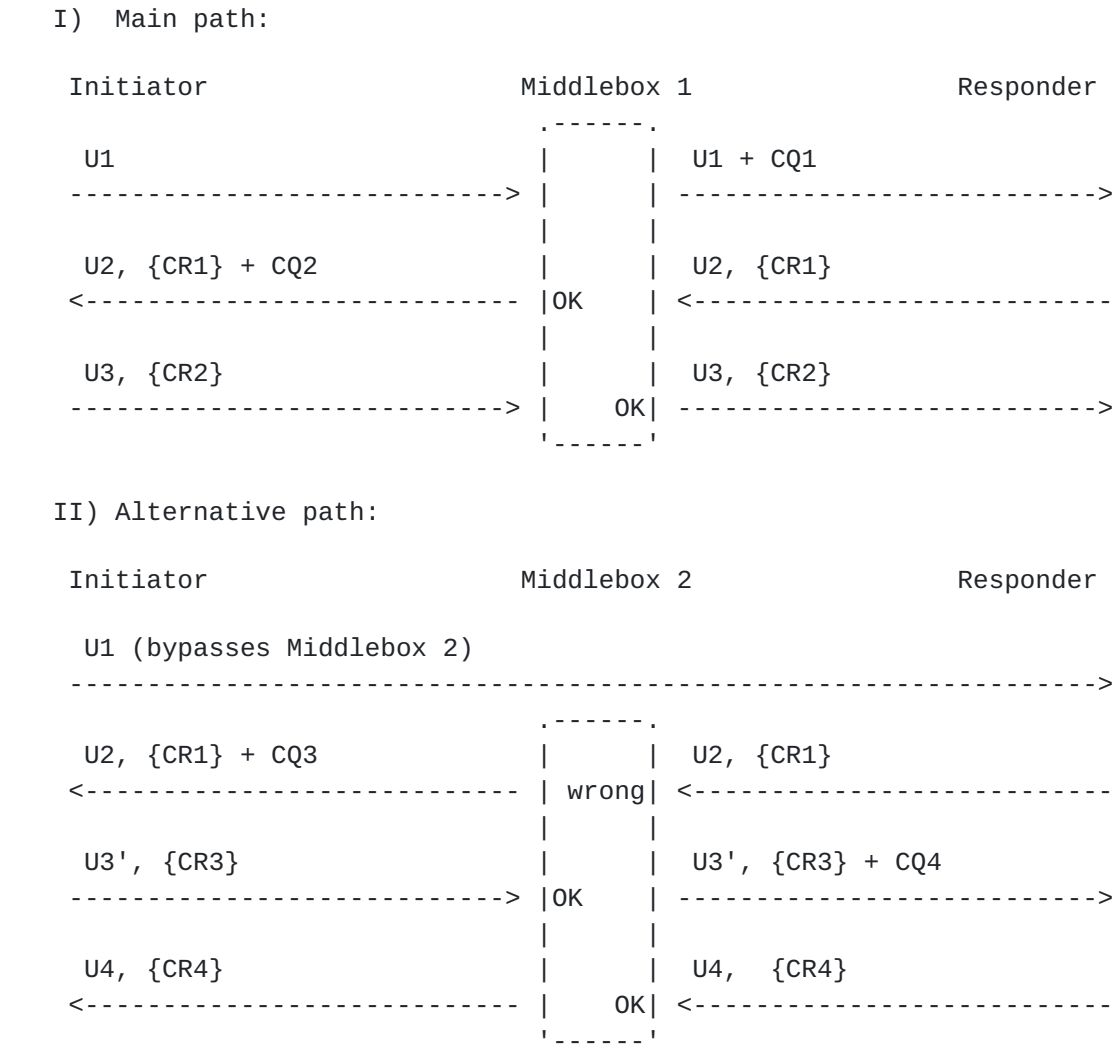
Figure 2

Middlebox 1 in Figure 2 can verify the identity of the Responder by checking its PK signature and the presence of the CHALLENGE_RESPONSE in the U2 packet. If necessary, the middlebox MAY add an CHALLENGE_REQUEST for the Initiator of the update. The middlebox can verify the Initiator's identity by verifying its signature and the CHALLENGE_RESPONSE in the U3 packet.

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 of the hosts or not, the middlebox forwards different packets and, thus, needs to interact differently with the updates. Figure 3 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 CHALLENGE_RESPONSE (CR1) in the second UPDATE packet. Thus, if a middlebox encounters non-matching or missing CHALLENGE_RESPONSE parameter in an initial update packet, the middlebox SHOULD ignore it.

Complying to the RESPONSE RULE stated in Section Section 2.2.2, the RESPONDER generates an additional fourth update packet on receiving the CHALLENGE_REQUEST. 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 3.



Middlebox authentication of a HIP mobility update over different paths.

Figure 3

2.2.4. Identity Signaling During Updates

As middleboxes have to verify rapidly and forward HIP packets, they need to be supplied with all information necessary to do so. If end-hosts hand over communication to a new communication path, middleboxes need to be able to learn their Host Identifiers (HIs) from the UPDATE packets. Therefore, all packets that contain a CHALLENGE_RESPONSE parameter MUST contain the HOST_ID parameter.

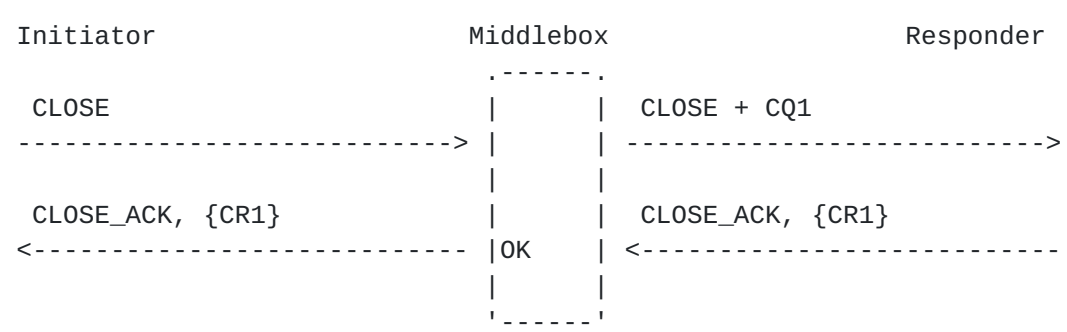
2.2.5. Closing of Connections

The connection tear down as defined in [RFC5201] consists of two consecutive messages. This lack of a third message restricts middleboxes to authenticating the Responder of a CLOSE packet. However, verifying the legitimacy of the Responder suffices in most network scenarios, as CLOSE packets from unauthentic Initiators will be dropped by the Responder due to an invalid HMAC parameter. As a result, on-path middleboxes will not see CLOSE_ACK packets for rejected CLOSE packets. CLOSE_ACK packets can be authenticated by the middleboxes by adding a CHALLENGE_REQUEST parameter to the corresponding CLOSE packet as described above. Hence, middleboxes do not falsely tear down connections on illegitimate (forged) CLOSE packets.

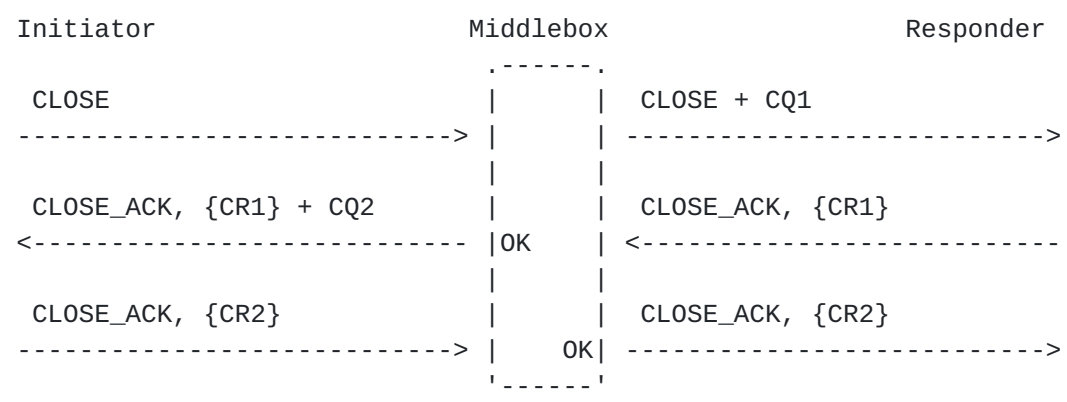
If local policies still require a middlebox to authenticate the CLOSE messages of both peers, the tear down operation needs to be extended following the RESPONSE RULE in Section 2.2.2. Hence, the responder side CLOSE_ACK packet MUST be followed by an initiator side CLOSE_ACK if the received CLOSE_ACK packet contains a CHALLENGE_REQUEST parameter.

Middleboxes should have learned the identities of the peers during the BEX or an UPDATE prior to the CLOSE exchange. Hence, end-hosts are not required to include their identities in the CLOSE exchange. If a middlebox has not learned the identities of the peers when inspecting a CLOSE packet, it MUST forward the packet. In order to prevent misuse of the CLOSE exchange as a side channel for disallowed communication, middleboxes SHOULD rate limit unauthenticated CLOSE exchanges.

I) Regular CLOSE authentication:



II) Extended CLOSE authentication:



CQ: Middlebox challenge reQuest
CR: Middlebox challenge Response
{}: Signature with sender's HI as key

Middlebox authentication of a HIP close with authentication of (I) the Responder and (II) both peers.

Figure 4

2.3. Failure Signaling

Middleboxes SHOULD inform the sender of a BEX packet or update packet if it does not satisfy the requirements of the middlebox. Reasons for non-satisfactory packets are missing HOST_ID or CHALLENGE_RESPONSE parameters. Other reasons may be middlebox policies regarding, for example, insufficient client capabilities or or insufficient credentials delivered in a HIP CERT parameter [RFC6253]. 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

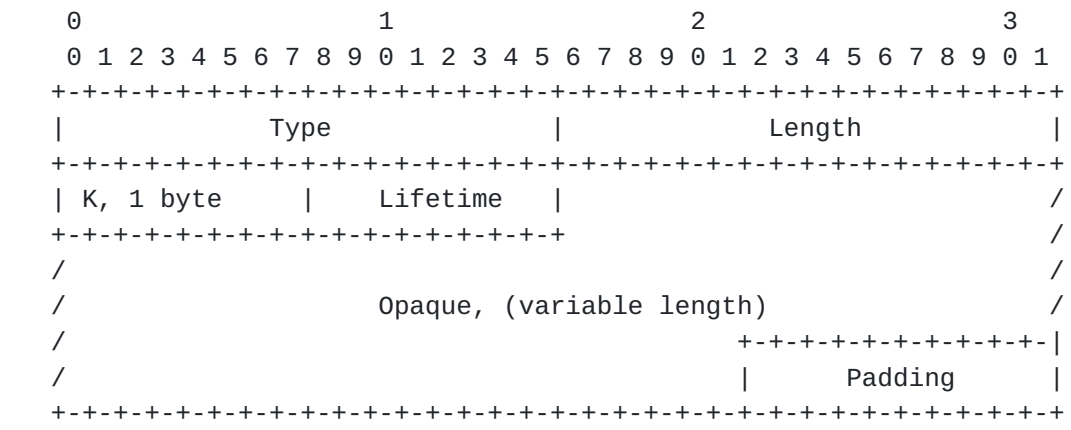
Analogously to the specification in [RFC5201], HIP aware middleboxes SHOULD support IP-level fragmentation and reassembly for IPv6 and MUST support IP-level fragmentation and reassembly for IPv4. However, when adding CHALLENGE_REQUEST parameters, a middlebox SHOULD keep the total packet size below 1280 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. CHALLENGE_REQUEST

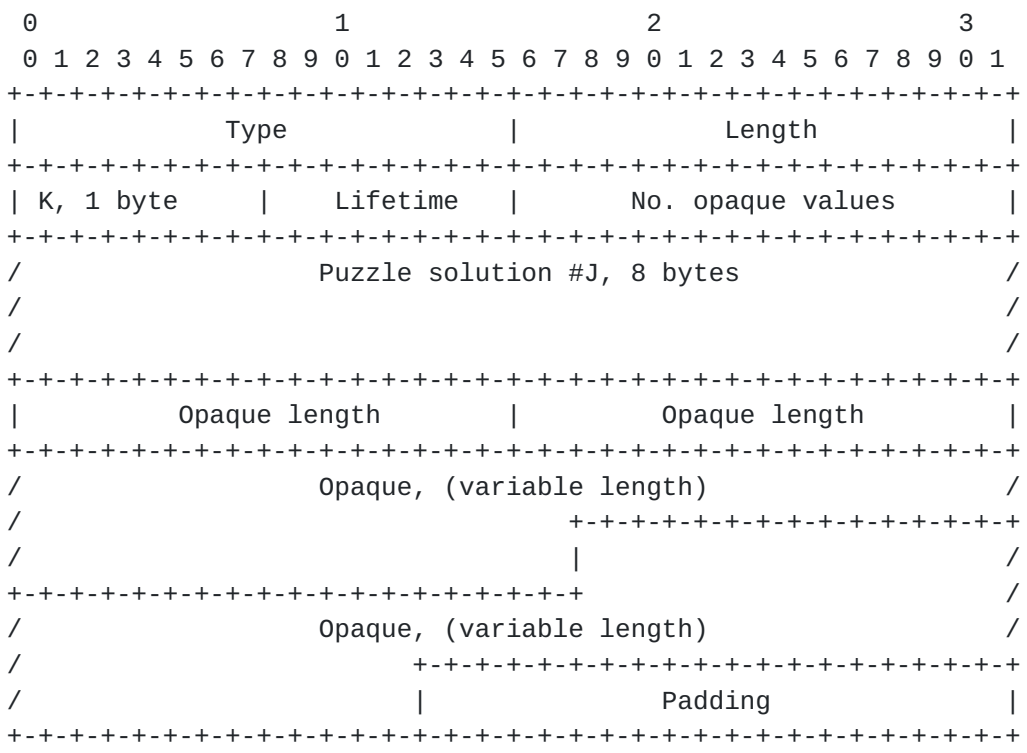
A middlebox MAY append the CHALLENGE_REQUEST parameter to R1, I2, and UPDATE packets. The structure of the CHALLENGE_REQUEST parameter is depicted in the following figure. The semantics of the K and Lifetime fields are identical to the fields defined in the PUZZLE parameter in [RFC5201]. The opaque data field serves as nonce and puzzle seed value. To generate the seed corresponding to the 8-byte value I in [RFC5201], the receiver of the puzzle applies Ltrunc as defined in [RFC5201] to the received opaque data and truncates the result to 8 bytes. Note that the opaque data field must provide sufficient randomness to serve as puzzle seed.



Type	65334
Length	Variable
K	K is the number of verified bits
Lifetime	Challenge lifetime 2^(value-32) seconds
Opaque	Opaque data that serves as nonce and as basis for the puzzle. The puzzle value I is generated by hashing the opaque data field with the hash function SHA-1 and truncating it to 8-byte length.

2.5.2. CHALLENGE_RESPONSE

The CHALLENGE_RESPONSE parameter is the response to one or more CHALLENGE_REQUEST parameters. The receiver of a CHALLENGE_REQUEST parameter SHOULD reply with a CHALLENGE_RESPONSE. Otherwise, the middlebox that added the CHALLENGE_REQUEST parameter MAY decide to degrade or deny its service. The Opaque fields of the received CHALLENGE_REQUEST parameters must be copied to the CHALLENGE_RESPONSE parameter in the reverse order of reception without any modification. As the number of opaque fields may be variable, it is encoded in the CHALLENGE_RESPONSE parameter. Furthermore, the length of each Opaque value is variable and is included in the parameter. The Opaque values are appended behind the last Opaque length field. Instead of copying the Opaque field of each CHALLENGE_REQUEST parameter, the input for the puzzle generation procedure may be reused. If the puzzle difficulty in the received CHALLENGE_REQUEST parameters is set to any other value except 0, an appropriate puzzle solution (adhering to the SOLUTION specifications in [RFC5201]) must be provided in the CHALLENGE_RESPONSE parameter. The CHALLENGE_RESPONSE parameter is non-critical and covered by the SIGNATURE. The structure of the CHALLENGE_RESPONSE parameter is depicted below:



Type	322
Length	Variable
K	K is the number of verified bits
Lifetime	Challenge lifetime 2^(value-32) seconds
No. opaque values	Number of included opaque values
Puzzle solution	Random number
Opaque length	Length of an included Opaque field
Opaque	Copied unmodified from the received CHALLENGE_REQUEST parameters

3. Security Services for the HIP Control Channel

In this section, we define the adversary 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 compromise 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 [RFC5202] is currently the default payload encapsulation format for HIP. A limitation of ESP is that it does not provide a secure binding between the HIP control channel and the ESP traffic on a per-packet basis. Hence, the achievable level of security for the payload channel is lower compared to the HIP control channel.

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 is involved in the establishment of a HIP association and, thus, involved 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 forged 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 an 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. Also, 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 is unable to 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 and out of scope of this document.

5. Security Considerations

This HIP extension specifies how HIP-aware middleboxes interact with the handshake and mobility-signaling of the Host Identity Protocol. The scope is restricted to the authentication of end-hosts and excludes the issue of stronger authentication of 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.

6. IANA Considerations

This document specifies two new HIP parameter types. The preliminary parameter type numbers are 322 and 65334.

7. Acknowledgments

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

8. Changelog

8.1. Version 4

- Some clarifications.

- Add new way to compute single solution for multiple CHALLENGE_REQUEST parameters.

- Modify parameter layout for CHALLENGE_RESPONSE parameter.

- Add middlebox authentication for the CLOSE exchange.

- Updated outdated references.

8.2. Version 3

- Some editorial changes.

- Added text about space issues in response packets with too many CHALLENGE_RESPONSE parameters in Section Section 2.1.2

9. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

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

[RFC5202] Jokela, P., Moskowitz, R., and P. Nikander, "Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP)", RFC 5202, April 2008.

[RFC5203] Laganier, J., Koponen, T., and L. Eggert, "Host Identity Protocol (HIP) Registration Extension", RFC 5203, April 2008.

[RFC5206] Nikander, P., Henderson, T., Vogt, C., and J. Arkko, "End-Host Mobility and Multihoming with the Host Identity Protocol", RFC 5206, April 2008.

[RFC5207] Stiemerling, M., Quittek, J., and L. Eggert, "NAT and Firewall Traversal Issues of Host Identity Protocol (HIP) Communication", RFC 5207, April 2008.

[RFC5770] Komu, M., Henderson, T., Tschofenig, H., Melen, J., and A. Keranen, "Basic Host Identity Protocol (HIP) Extensions for Traversal of Network Address Translators", RFC 5770, April 2010.

[RFC6253] Heer, T. and S. Varjonen, "Host Identity Protocol Certificates", RFC 6253, May 2011.

Authors' Addresses

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

Email: heer@cs.rwth-aachen.de
URI: http://www.comsys.rwth-aachen.de/team/tobias-heer/

Rene Hummen
RWTH Aachen University, Communication and Distributed Systems Group
Ahornstrasse 55
Aachen 52062
Germany

Email: hummen@cs.rwth-aachen.de
URI: http://www.comsys.rwth-aachen.de/team/rene-hummen/

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

Email: heer@cs.rwth-aachen.de
URI: http://www.comsys.rwth-aachen.de/team/klaus/

Miika Komu
Aalto University, Department of Computer Science and Engineering
Konemiehentie 2
Espoo
Finland

Phone: +358947027117
Fax: +358947025014
Email: miika@iki.fi
URI: http://www.hiit.fi/