Internet Engineering Task Force (IETF)
Request for Comments: 6078
Category: Experimental
ISSN: 2070-1721

G. Camarillo
J. Melen
Ericsson

January 2011

Host Identity Protocol (HIP) Immediate Carriage and Conveyance of Upper-Layer Protocol Signaling (HICCUPS)

Abstract

This document defines a new Host Identity Protocol (HIP) packet type called DATA. HIP DATA packets are used to reliably convey authenticated arbitrary protocol messages over various overlay networks.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for examination, experimental implementation, and evaluation.

This document defines an Experimental Protocol for the Internet community. This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6078.

Table of Contents

1. Introduction 3
2. Terminology 3
3. Background on HIP 4
3.1. Message Formats 4
3.1.1. HIP Fixed Header 4
3.1.2. HIP Parameter Format 5
3.2. HIP Base Exchange, Updates, and State Removal 5
4. Definition of the HIP_DATA Packet 6
4.1. Definition of the SEQ_DATA Parameter 8
4.2. Definition of the ACK_DATA Parameter 8
4.3. Definition of the PAYLOAD_MIC Parameter 9
4.4. Definition of the TRANSACTION_ID Parameter 10
5. Generation and Reception of HIP_DATA Packets 10
5.1. Handling of SEQ_DATA and ACK_DATA 10
5.2. Generation of a HIP_DATA Packet 11
5.3. Reception of a HIP_DATA Packet 12
5.3.1. Handling of SEQ_DATA in a Received HIP_DATA Packet . . 13
5.3.2. Handling of ACK_DATA in a Received HIP_DATA Packet . . 14
6. Use of the HIP_DATA Packet 14
7. Security Considerations 15
8. IANA Considerations 16
9. Acknowledgments 16
10. References 16
10.1. Normative References 16
10.2. Informative references 16

1. Introduction

Two hosts can use HIP [RFC5201] to establish a security association (SA) between them in order to exchange arbitrary protocol messages over that security association. The establishment of such a security association involves a four-way handshake referred to as the HIP base exchange. When handling communications between the hosts, HIP supports mobility, multihoming, security, and NAT traversal. Some applications require these features for their communications but cannot accept the overhead involved in establishing a security association (i.e., the HIP base exchange) before those communications can start.

In this document, we define the HIP DATA packet, which can be used to convey (in a authenticated and reliable way) protocol messages to a remote host without running the HIP base exchange. The HIP_DATA packet has the following semantics: unordered, duplicate free, reliable, and authenticated message-based delivery service. We also discuss the trade-offs involved in using this packet (i.e., less overhead but also less denial-of-service (DoS) protection) and the situations where it is appropriate to use this packet. The HIP_DATA packet is not intended to be a replacement for the Encapsulating Security Payload (ESP) transport; instead, it SHOULD NOT be used to exchange more than a few packets between peers. If a continuous communication is required or communication that requires confidentiality protection then hosts MUST run the HIP base exchange to set up an ESP security association. Additionally, APIs to higher-level protocols that might use this service are outside of the scope of this document.

2. Terminology

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 [RFC2119]. In addition, this document uses the terms defined in [RFC5201].

Message Integrity Code (MIC) is a collision-resistant hash sum calculated over the message that is being integrity protected. The MIC does not use secret keys, and thus it needs additional means to ensure that it has not been tampered with during transmission. Essentially, the MIC is same as the Message Authentication Code (MAC) with the distinction that the MIC does not use secret keys. The MIC is also often referred as the Integrity Check Value (ICV), fingerprint, or unkeyed MAC.

3. Background on HIP

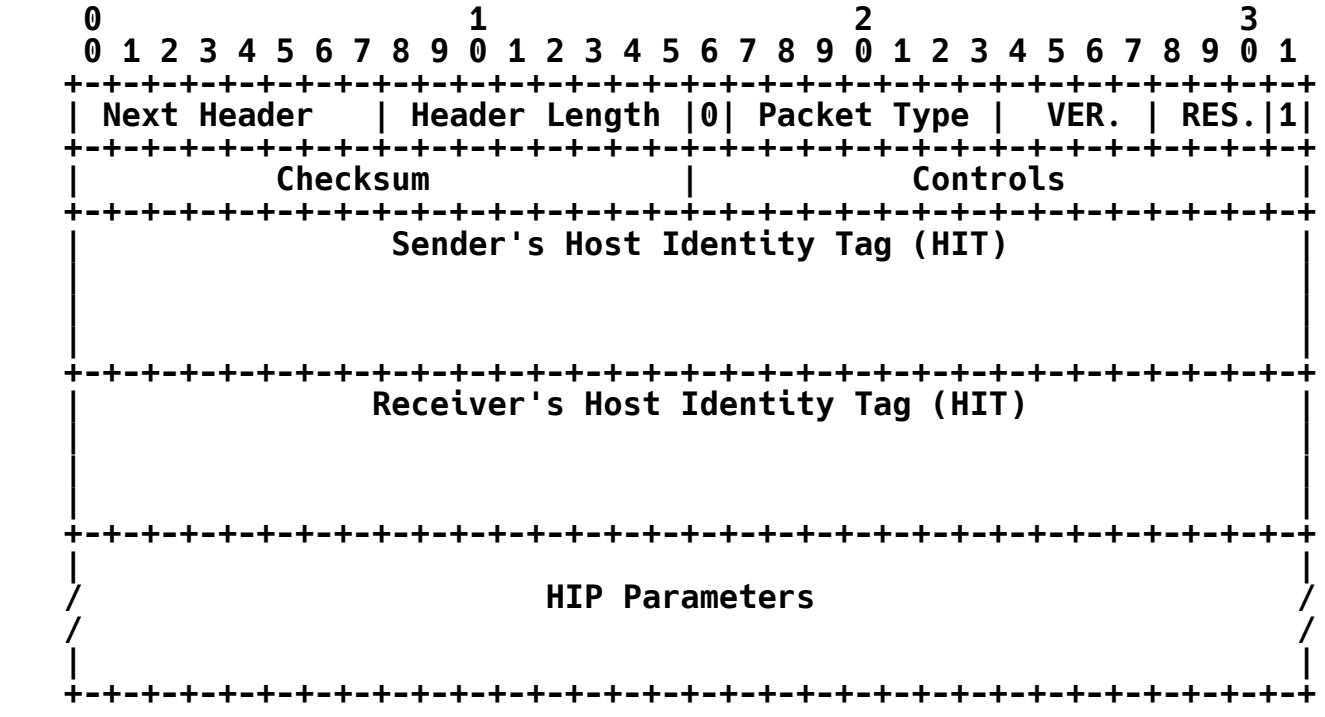
The HIP specification [RFC5201] defines a number of messages and parameters. The parameters are encoded as TLVs, as shown in Section 3.1.2. Furthermore, the HIP header carries a Next Header field, allowing other arbitrary packets to be carried within HIP packets.

3.1. Message Formats

3.1.1. HIP Fixed Header

The HIP packet format consists of a fixed header followed by a variable number of parameters. The parameter format is described in Section 3.1.2.

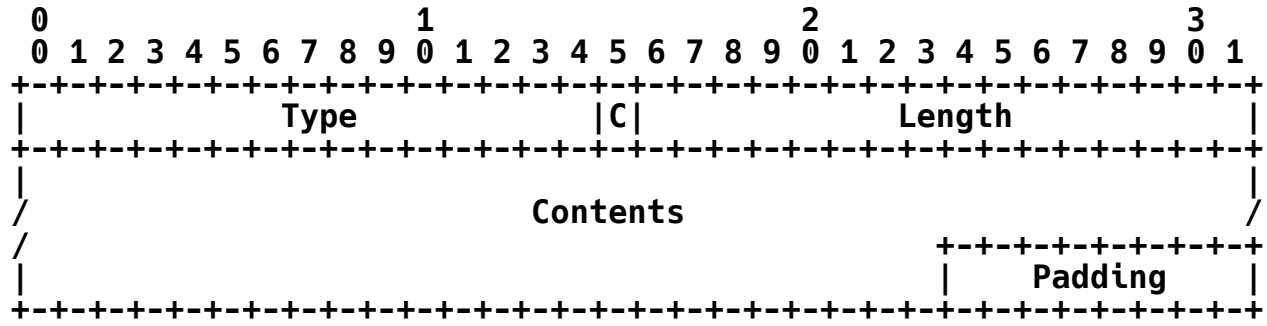
The fixed header is defined in Section 5.1 of [RFC5201] and copied below.



The HIP header is logically an IPv6 extension header. The HIP specification [RFC5201] defines handling only for Next Header value decimal 59, IPv6-NoNxt [PROTOCOL-NUMBERS], the IPv6 'no next header' value. This document describes processing for Next Header values other than decimal 59, which indicates that there are either more extension headers and/or data following the HIP header.

3.1.2. HIP Parameter Format

The HIP parameter format is defined in Section 5.2.1 of [RFC5201], and copied below.



Type	Type code for the parameter. 16 bits long, C-bit being part of the Type code.
C	Critical. One if this parameter is critical, and MUST be recognized by the recipient; zero otherwise. The C bit is considered to be a part of the Type field. Consequently, critical parameters are always odd and non-critical ones have an even value.
Length	Length of the Contents, in octets.
Contents	Parameter specific, defined by Type.
Padding	Padding, 0-7 octets, added if needed.

3.2. HIP Base Exchange, Updates, and State Removal

The HIP base exchange is a four-message authentication and key exchange protocol that creates shared, mutually authenticated keying material at the communicating parties. These keying materials, together with associated public keys and IP addresses, form a HIP security association (SA). The details of the protocol are defined in the HIP base exchange specification [RFC5201].

In addition to creating the HIP SA, the base exchange messages may carry additional parameters that are used to create additional state. For example, the HIP ESP specification [RFC5202] defines how HIP can be used to create end-to-end, host-to-host IPsec ESP security associations, used to carry data packets. However, it is important to understand that the HIP base exchange is by no means bound to IPsec; using IPsec ESP to carry data traffic forms just a baseline and ensures interoperability between initial HIP implementations.

Once there is a HIP SA between two HIP-enabled hosts, they can exchange further HIP control messages. Typically, UPDATE messages are used. For example, the HIP mobility and multihoming specification [RFC5206] defines how to use UPDATE messages to change the set of IP addresses associated with a HIP SA.

In addition to the base exchange and updates, the HIP base protocol specification also defines how one can remove a HIP SA once it is no longer needed.

4. Definition of the HIP_DATA Packet

The HIP DATA packet can be used to convey protocol messages to a remote host without running the HIP base exchange. HIP DATA packets are transmitted reliably, as discussed in Section 5. The payload of a HIP_DATA packet is placed after the HIP header and protected by a PAYLOAD_MIC parameter, which is defined in Section 4.3. The following is the definition of the HIP_DATA packet (see the definition of notation in [RFC5201], Section 2.2):

```
   Header:
     Packet Type = 32
     SRC HIT = Sender's HIT
     DST HIT = Receiver's HIT

 IP ( HIP ( [HOST_ID, ] SEQ_DATA, PAYLOAD_MIC,  [ PAYLOAD_MIC, ..., ]
            HIP_SIGNATURE) PAYLOAD )

 IP ( HIP ( [HOST_ID, ] SEQ_DATA, ACK_DATA, PAYLOAD_MIC,
            [ PAYLOAD_MIC, ..., ] HIP_SIGNATURE) PAYLOAD )

 IP ( HIP ( [HOST_ID, ] ACK_DATA, HIP_SIGNATURE))
```

The SEQ_DATA and ACK_DATA parameters are defined in Sections 4.1 and 4.2, respectively. They are used to provide a reliable delivery of HIP_DATA packets, as discussed in Section 5.

The HOST_ID parameter is defined in Section 5.2.8 of [RFC5201]. This parameter is the sender's Host Identifier that is used to compute the HIP_DATA packet's signature and to verify it against the received signature. The HOST_ID parameter is optional as it MAY have been delivered using out-of-band mechanism to the receiver. If the host doesn't have reliable information that the corresponding node has its HOST_ID, it MUST always include the HOST_ID in the packet. If the receiver is unable to verify the SIGNATURE, then the packet MUST be dropped and the appropriate NOTIFY packet SHOULD be sent to the sender indicating AUTHENTICATION_FAILED as described in [RFC5201], Section 5.2.16.

The PAYLOAD_MIC parameter is defined in Section 4.3. This parameter contains the MIC of the payload carried by the HIP_DATA packet. The PAYLOAD_MIC contains the collision-resistant hash of the payload following the HIP DATA. The PAYLOAD_MIC is included in the signed part of the HIP DATA packet and gives integrity protection for the packet as well as the payload carried after it.

The HIP_SIGNATURE parameter is defined in Section 5.2.11 of [RFC5201]. It contains a signature over the contents of the HIP_DATA packet. The calculation and verification of the signature is defined in Section 6.4.2. of [RFC5201].

Section 5.3 of [RFC5201] states the following:

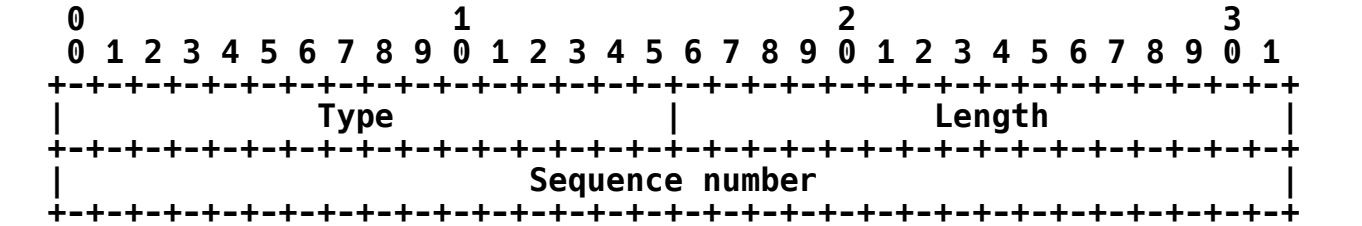
> In the future, an OPTIONAL upper-layer payload MAY follow the HIP header. The Next Header field in the header indicates if there is additional data following the HIP header.

We have chosen to place the payload after the HIP extension header and only to place a MIC of the payload into the HIP extension header in a PAYLOAD_MIC parameter because that way the data integrity is protected by a public key signature with the help of the MIC. The payload that is protected by the PAYLOAD_MIC parameter has been linked to the appropriate upper-layer protocol by storing the upper-layer protocol number, 8 octets of payload data, and by calculating a hash sum (MIC) over the data. The HIP_DATA packet MAY contain one or more PAYLOAD_MIC parameters, each bound to a different Next Header type. The hash algorithm used to generate the MIC is the same as the algorithm used to generate the Host Identity Tag [RFC5201].

Upper-layer protocol messages, such as overlay network control traffic, sent in HIP DATA messages may need to be matched to different transactions. For this purpose, a DATA message MAY also contain a TRANSACTION_ID parameter. The identifier value is a variable length bit string in network byte order that is unique for each transaction. A response to a request uses the same identifier value, thereby allowing the receiver to match requests to responses.

4.1. Definition of the SEQ_DATA Parameter

The following is the definition of the SEQ_DATA parameter:

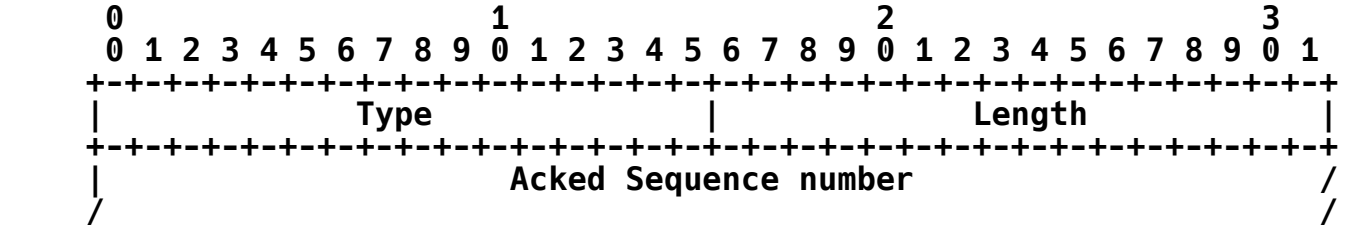


Type	4481
Length	4
Sequence number	32-bit unsigned integer in network byte order that MUST NOT be reused before it has been acknowledged by the receiver.

This parameter has the critical bit set. If it is not supported by the receiver, the packet MUST be dropped and the appropriate NOTIFY packet SHOULD be sent to the sender indicating UNSUPPORTED_CRITICAL_PARAMETER_TYPE as described in [RFC5201], Section 5.2.16.

4.2. Definition of the ACK_DATA Parameter

The following is the definition of the ACK_DATA parameter:

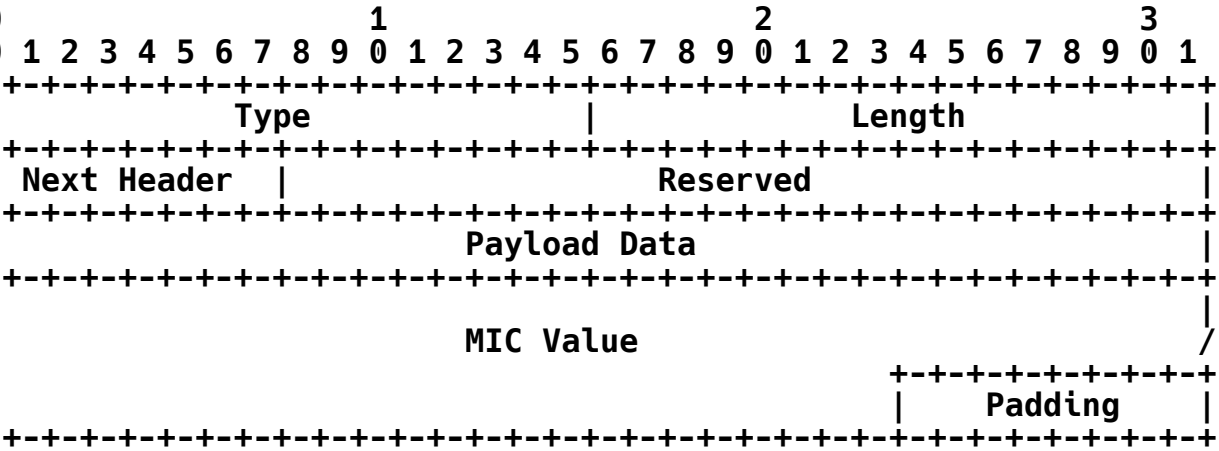


Type	4545
Length	variable (multiple of 4)
Acked Sequence number	A sequence of 32-bit unsigned integers in network byte order corresponding to the sequence numbers being acknowledged.

This parameter has the critical bit set. If it is not supported by the receiver, the packet MUST be dropped and the appropriate NOTIFY packet SHOULD be sent to the sender indicating UNSUPPORTED_CRITICAL_PARAMETER_TYPE as described in [RFC5201], Section 5.2.16.

4.3. Definition of the PAYLOAD_MIC Parameter

The following is the definition of the PAYLOAD_MIC parameter:



Type	4577
Length	Length in octets, excluding Type, Length, and Padding.
Next Header	Identifies the data that is protected by this MIC. The values for this field are defined by IANA "Protocol Numbers" [PROTOCOL-NUMBERS].
Payload Data	Last 8 octets of the payload data over which the MIC is calculated. This field is used to uniquely bind the PAYLOAD_MIC parameter to the Next Header, in case there are multiple copies of the same type.
MIC Value	MIC computed over the data to which the Next Header and Payload Data point. The size of the MIC is the natural size of the computation output depending on the function used.

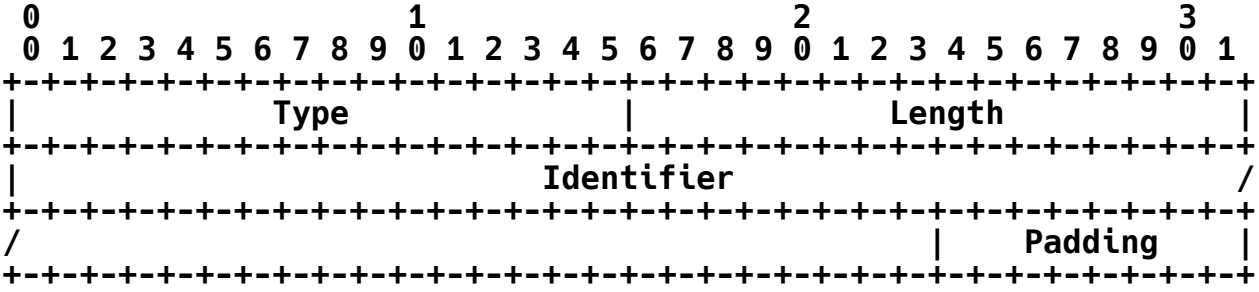
This parameter has the critical bit set. If it is not supported by the receiver, the packet MUST be dropped and the appropriate NOTIFY packet SHOULD be sent to the sender indicating UNSUPPORTED_CRITICAL_PARAMETER_TYPE as described in [RFC5201], Section 5.2.16.

There is a theoretical possibility that when generating multiple PAYLOAD_MIC parameters that will be carried in a single packet, they would have identical Next Header and Payload Data fields; thus, it is required that PAYLOAD_MIC parameters MUST follow the natural order of extension headers in the packet so that it's possible to bind PAYLOAD_MICs to correct payload data. In case the receiving host is still unable to identify the payloads, it MUST drop the packet and

SHOULD send a NOTIFY packet to the sender indicating INVALID_SYNTAX as described in [RFC5201], Section 5.2.16.

4.4. Definition of the TRANSACTION_ID Parameter

The following is the definition of the TRANSACTION_ID parameter:



Type	4580
Length	Length of the Identifier, in octets
Identifier	The identifier value
Padding	0-7 octets of padding if needed

Figure 1

5. Generation and Reception of HIP_DATA Packets

HIP_DATA packets are transmitted reliably. Reliable delivery is achieved through the use of retransmissions and of the SEQ_DATA and ACK_DATA parameters.

5.1. Handling of SEQ_DATA and ACK_DATA

A HIP_DATA packet MUST contain at least one of a SEQ_DATA or an ACK_DATA parameter; if both parameters are missing, then packet MUST be dropped as invalid.

A HIP_DATA packet containing a SEQ_DATA parameter MUST contain one or more PAYLOAD_MIC parameters; otherwise, the packet MUST be dropped. The presence of a SEQ_DATA parameter indicates that the receiver MUST ACK the HIP_DATA packet. A HIP_DATA packet that does not contain a SEQ_DATA parameter is simply an ACK of a previous HIP_DATA packet, and it MUST NOT be ACKed.

A HIP_DATA packet containing an ACK_DATA parameter echoes the SEQ_DATA sequence numbers of the HIP_DATA packets being acknowledged. The ACK_DATA parameter MUST acknowledge at least one SEQ_DATA sequence number and MAY acknowledge multiple SEQ_DATA sequence numbers by adding all of them to the ACK_DATA parameter.

A HIP_DATA packet MAY contain both a SEQ_DATA and an ACK_DATA parameter. In this case, the ACK is being piggybacked on an outgoing HIP_DATA packet. In general, HIP_DATA packets carrying SEQ_DATA SHOULD be ACKed upon completion of the processing of the HIP_DATA packet. A host MAY choose to hold the HIP DATA packet carrying an ACK for a short period of time to allow for the possibility of piggybacking the ACK_DATA parameter, in a manner similar to TCP delayed acknowledgments.

5.2. Generation of a HIP_DATA Packet

When a host has upper-layer protocol data to send, it either runs the HIP base exchange and sends the data over a SA, or sends the data directly using a HIP_DATA packet. Section 6 discusses when it is appropriate to use each method. This section discusses the case when the host chooses to use a HIP_DATA packet to send the upper-layer protocol data.

1. The host creates a HIP_DATA packet that contains a SEQ_DATA parameter. The host is free to choose any value for the SEQ_DATA sequence number in the first HIP_DATA packet it sends to a destination. After that first packet, the host MUST choose the value of the SEQ_DATA sequence number in subsequent HIP_DATA packets to the same destination so that no SEQ_DATA sequence number is reused before the receiver has closed the processing window for the previous packet using the same SEQ_DATA sequence number. Practically, giving the values of the retransmission timers used with HIP_DATA packets, this means that hosts must wait the maximum likely lifetime of the packet before reusing a given SEQ_DATA sequence number towards a given destination. However, it is not required for the node to know the maximum packet lifetime. Rather, it is assumed that the requirement can be met by maintaining the value as a simple, 32-bit, "wrap-around" counter, incremented each time a packet is sent. It is an implementation choice whether to maintain a single counter for the node or multiple counters (one for each <source, destination> HIT pair).

2. The host creates the PAYLOAD_MIC parameter. The MIC is a hash calculated over the whole PAYLOAD that the Next Header field of the PAYLOAD_MIC parameter indicates. If there are multiple Next Header types that the host wants to protect, it SHOULD create separate PAYLOAD_MIC parameters for each of these. The receiver MUST validate all these MICs as described in Section 5.3.1. For calculating the MIC, the host MUST use the same hash algorithm as the one that has been used for generating the host's HIT as defined in Section 3.2. of [RFC5201].

3. The host creates the HIP_SIGNATURE parameter. The signature is calculated over the whole HIP envelope, excluding any parameters after the HIP_SIGNATURE, as defined in Section 5.2.11. of [RFC5201]. The receiver MUST validate this signature. It MAY use either the HI in the packet or the HI acquired by some other means.

4. The host sends the created HIP_DATA packet and starts a DATA timer. The default value for the timer is 3 seconds. If multiple HIP DATA packets are outstanding, multiple timers are in effect.

5. If the DATA timer expires, the HIP_DATA packet is resent. The HIP DATA packet can be resent DATA_RETRY_MAX times. The DATA timer MUST be exponentially backed off for subsequent retransmissions. If no acknowledgment is received from the peer after DATA_RETRY_MAX times, the delivery of the HIP_DATA packet is considered unsuccessful and the application is notified about the error. The DATA timer is canceled upon receiving an ACK from the peer that acknowledges receipt of the HIP_DATA packet. The default value for DATA_RETRY_MAX SHOULD be 5 retries, but it MAY be changed through local policy.

5.3. Reception of a HIP_DATA Packet

A host receiving a HIP_DATA packet makes a decision whether or not to process the packet. If the host, following its local policy, suspects that this packet could be part of a DoS attack. The host MAY respond with an R1 packet to the HIP_DATA packet, if the packet contained SEQ_DATA and PAYLOAD_MIC parameters, in order to indicate that HIP base exchange MUST be completed before accepting payload packets from the originator of the HIP_DATA packet.

From RFC 5201 (Section 4.1):

> The HIP base exchange serves to manage the establishment of state between an Initiator and a Responder. The first packet, I1, initiates the exchange, and the last three packets, R1, I2, and R2, constitute an authenticated Diffie-Hellman [DIF76] key exchange for session key generation.

If the host chooses to respond to the HIP DATA with an R1 packet, it creates a new R1 or selects a precomputed R1 according to the format described in [RFC5201], Section 5.3.2. The host SHOULD drop the received data packet if it responded with an R1 packet to the HIP_DATA packet. The sender of HIP_DATA packet is responsible for retransmission of the upper-layer protocol data after successful completion of the HIP base exchange.

If the host, following its local policy, decides to process the incoming HIP_DATA packet, it processes the packet according to the following rules:

1. If the HIP_DATA packet contains a SEQ_DATA parameter and no ACK_DATA parameter, the HIP_DATA packet is processed and replied to as described in Section 5.3.1.

2. If the HIP_DATA packet contains an ACK_DATA parameter and no SEQ_DATA parameter, the HIP_DATA packet is processed as described in Section 5.3.2.

3. If the HIP_DATA packet contains both a SEQ_DATA parameter and an ACK_DATA parameter, the HIP_DATA packet is processed first as described in Section 5.3.2, and then the rest of the HIP_DATA packet is processed and replied to as described in Section 5.3.1.

5.3.1. Handling of SEQ_DATA in a Received HIP_DATA Packet

The following steps define the conceptual processing rules for handling a SEQ_DATA parameter in a received HIP_DATA packet.

The system MUST verify the SIGNATURE in the HIP_DATA packet. If the verification fails, the packet SHOULD be dropped and an error message logged.

If the value in the received SEQ_DATA and the MIC value in the received PAYLOAD_MIC correspond to a HIP_DATA packet that has recently been processed, the packet is treated as a retransmission. It is recommended that a host cache HIP_DATA packets with ACKs to avoid the cost of generating a new ACK packet to respond to a retransmitted HIP_DATA packet. The host MUST acknowledge, again, such (apparent) HIP_DATA packet retransmissions but SHOULD also consider rate-limiting such retransmission responses to guard against replay attacks.

The system MUST verify the PAYLOAD_MIC by calculating the MIC over the PAYLOAD that the Next Header field indicates. For calculating the MIC, the host will use the same hash algorithm that has been used to generate the sender's HIT as defined in Section 3.2. of [RFC5201]. If the packet carried multiple PAYLOAD_MIC parameters, each of them are verified as described above. If one or more of the verifications fail, the packet SHOULD be dropped and an error message logged.

If a new SEQ parameter is being processed, the parameters in the HIP DATA packet are then processed.

A HIP_DATA packet with an ACK_DATA parameter is prepared and sent to the peer. This ACK_DATA parameter may be included in a separate HIP DATA packet or piggybacked in a HIP_DATA packet with a SEQ_DATA parameter. The ACK_DATA parameter MAY acknowledge more than one of the peer's HIP_DATA packets.

5.3.2. Handling of ACK_DATA in a Received HIP_DATA Packet

The following steps define the conceptual processing rules for handling an ACK_DATA parameter in a received HIP_DATA packet.

The system MUST verify the SIGNATURE in the HIP_DATA packet. If the verification fails, the packet SHOULD be dropped and an error message logged.

The sequence numbers reported in the ACK_DATA must match with a previously sent HIP_DATA packet containing SEQ_DATA that has not already been acknowledged. If no match is found or if the ACK_DATA does not acknowledge a new HIP_DATA packet, the packet either MUST be dropped if no SEQ_DATA parameter is present or the processing steps in Section 5.3.1 are followed.

The corresponding DATA timer is stopped so that the now acknowledged HIP_DATA packet is no longer retransmitted. If multiple HIP_DATA packets are newly acknowledged, multiple timers are stopped.

6. Use of the HIP_DATA Packet

HIP currently requires that the four-message base exchange is executed at the first encounter of hosts that have not communicated before. This may add additional RTTs (Round-Trip Times) to protocols based on a single message exchange. However, the four-message exchange is essential to preserve the DoS protection nature of the base exchange. The use of the HIP_DATA packet defined in this document reduces the initial overhead in the communications between two hosts. However, the HIP_DATA packet itself does not provide any protection against DoS attacks. Therefore, the HIP_DATA packet MUST only be used in environments whose policies provide protection against DoS attacks. For example, a HIP-based overlay may have policies in place to control which nodes can join the overlay. However, authorization of who is allowed to join the overlay is beyond the scope of this specification. Any particular node in the overlay may want to accept HIP_DATA packets from other nodes in the overlay, given that those other nodes were authorized to join the overlay. However, the same node will not accept HIP_DATA packets from random nodes that are not part of the overlay. Additionally, the HIP_DATA packet itself does not provide confidentiality for its payload. Therefore, the HIP_DATA packet MUST NOT be used in

environments that do not provide an appropriate level of confidentiality (e.g., a HIP-based overlay MUST NOT send HIP_DATA packets unless the connections between overlay nodes are encrypted).

The type of data to be sent is also relevant to whether the use of a HIP_DATA packet is appropriate. HIP itself does not support fragmentation but relies on underlying IP-layer fragmentation. This may lead to reliability problems in the case where a message cannot be easily split over multiple HIP messages. Therefore, applications in environments where fragmentation could be an issue SHOULD NOT generate large HIP_DATA packets that may lead to fragmentation. The implementation SHOULD check the MTU of the link before sending the packet, and if the packet size is larger than MTU, it SHOULD signal to the upper-layer protocol if the packet results in an ICMP error message. Note that there are environments where fragmentation is not an issue. For example, in some HIP-based overlays, nodes can exchange HIP_DATA packets on top of TCP connections that provide transport-level fragmentation and, thus, avoid IP-level fragmentation.

HIP currently requires that all messages excluding I1s but including HIP_DATA packets are digitally signed. This adds to the packet size and the processing capacity needed to send packets. However, in applications where security is not paramount, it is possible to use very short keys, thereby reducing resource consumption.

7. Security Considerations

HIP is designed to provide secure authentication of hosts. HIP also attempts to limit the exposure of the host to various denial-of-service and man-in-the-middle (MitM) attacks. However, HIP_DATA packet, which can be sent without running the HIP base exchange between hosts has a trade-off that it does not provide the denial-of-service protection or confidentiality protection that HIP generally provides. Thus, the host should consider always situations where it is appropriate to send or receive HIP_DATA packet. If the communication consists more than few round trips of data or the data is highly sensitive in nature the host SHOULD run the base exchange with the peer host.

HIP_DATA packet is designed to protect hosts from second preimage attacks allowing receiving host to be able to detect, if the message was tampered during the transport. This property is also know as "weak collision-resistance". If a host tries to generate a second preimage, it would need to generate it such that the last 8 octets match with the original message.

When handling the PAYLOAD_MIC parameter in the receiving host, using the last 8 octets to identify the upper-layer protocol doesn't give any guarantee that the MIC would be correct; thus, an attacker could send packets where the next header and last 8 octets match the values carried by the PAYLOAD_MIC parameter. Therefore, it is always mandatory to verify the MIC value by calculating the hash over the payload.

8. IANA Considerations

This document updates the IANA registry for HIP packet types by introducing a new packet type for the HIP_DATA (Section 4) packet. This document updates the IANA registry for HIP parameter types by introducing new parameter values for the SEQ_DATA (Section 4.1), ACK_DATA (Section 4.2), PAYLOAD_MIC (Section 4.3), and TRANSACTION_ID (Section 4.4) parameters.

9. Acknowledgments

Pekka Nikander was one of the original authors of the document. Also, in the usual IETF fashion, a large number of people have contributed to the actual text or ideas. The list of these people include Miika Komu, Tobias Heer, Ari Keranen, Samu Varjonen, Thomas Henderson, and Jukka Ylitalo. Our apologies to anyone whose name is missing.

10. References

10.1. Normative References

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

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

[PROTOCOL-NUMBERS] IANA, "Protocol Numbers", <http://www.iana.org>.

10.2. Informative references

[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.

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

Authors' Addresses

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

EMail: Gonzalo.Camarillo@ericsson.com

Jan Melen
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

EMail: Jan.Melen@ericsson.com