HIP Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 28, 2017

A. Keranen
J. Melen
M. Komu, Ed.
Ericsson

March 27, 2017

Native NAT Traversal Mode for the Host Identity Protocol
draft-ietf-hip-native-nat-traversal-19

Abstract

This document specifies a new Network Address Translator (NAT) traversal mode for the Host Identity Protocol (HIP). The new mode is based on the Interactive Connectivity Establishment (ICE) methodology and UDP encapsulation of data and signaling traffic. The main difference from the previously specified modes is the use of HIP messages for all NAT traversal procedures.

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 September 28, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 3
2. Terminology 4
3. Overview of Operation 6
4. Protocol Description 7
4.1. Relay Registration 7
4.2. Transport Address Candidate Gathering 10
4.3. NAT Traversal Mode Negotiation 12
4.4. Connectivity Check Pacing Negotiation 13
4.5. Base Exchange via HIP Relay Server 14
4.6. Connectivity Checks 17
4.6.1. Connectivity Check Procedure 17
4.6.2. Rules for Connectivity Checks 20
4.6.3. Rules for Concluding Connectivity Checks 22
4.7. NAT Traversal Alternatives 23
4.7.1. Minimal NAT Traversal Support 23
4.7.2. Base Exchange without Connectivity Checks 23
4.7.3. Initiating a Base Exchange both with and without UDP Encapsulation 24
4.8. Sending Control Packets after the Base Exchange 25
4.9. Mobility Handover Procedure 26
4.10. NAT Keepalives 28
4.11. Closing Procedure 28
4.12. Relaying Considerations 29
4.12.1. Forwarding Rules and Permissions 29
4.12.2. HIP Data Relay and Relaying of Control Packets . . . 30
4.12.3. Handling Conflicting SPI Values 30
5. Packet Formats 31
5.1. HIP Control Packets 32
5.2. Connectivity Checks 32
5.3. Keepalives 33
5.4. NAT Traversal Mode Parameter 33
5.5. Connectivity Check Transaction Pacing Parameter 34
5.6. Relay and Registration Parameters 35
5.7. LOCATOR_SET Parameter 36
5.8. RELAY_HMAC Parameter 38
5.9. Registration Types 38
5.10. Notify Packet Types 38
5.11. ESP Data Packets 38
5.12. RELAYED_ADDRESS and MAPPED_ADDRESS Parameters 39
5.13. PEER_PERMISSION Parameter 40
5.14. HIP Connectivity Check Packets 41
5.15. NOMINATE parameter 42
6. Security Considerations 42

6.1. Privacy Considerations 42
6.2. Opportunistic Mode 43
6.3. Base Exchange Replay Protection for HIP Relay Server . . 43
6.4. Demultiplexing Different HIP Associations 43
6.5. Reuse of Ports at the Data Relay 44
6.6. Amplification attacks 44
6.7. Attacks against Connectivity Checks and Candidate Gathering 44
7. IANA Considerations 45
8. Contributors 45
9. Acknowledgments 45
10. References 46
10.1. Normative References 46
10.2. Informative References 47
Appendix A. Selecting a Value for Check Pacing 48
Appendix B. Differences with respect to ICE 49
Appendix C. Differences to Base Exchange and UPDATE procedures . 50
Authors' Addresses 52

1. Introduction

The Host Identity Protocol (HIP) [RFC7401] is specified to run directly on top of IPv4 or IPv6. However, many middleboxes found in the Internet, such as NATs and firewalls, often allow only UDP or TCP traffic to pass [RFC5207]. Also, especially NATs usually require the host behind a NAT to create a forwarding state in the NAT before other hosts outside of the NAT can contact the host behind the NAT. To overcome this problem, different methods, commonly referred to as NAT traversal techniques, have been developed.

The HIP experiment report [RFC6538] mentions that Teredo based NAT traversal for HIP and related ESP traffic (with double tunneling overhead). Two HIP specific NAT traversal techniques for HIP are specified in [RFC5770]. One of them uses only UDP encapsulation, while the other uses also the Interactive Connectivity Establishment (ICE) [I-D.ietf-ice-rfc5245bis] protocol, which in turn uses Session Traversal Utilities for NAT (STUN) [RFC5389] and Traversal Using Relays around NAT (TURN) [RFC5766] protocols to achieve a reliable NAT traversal solution.

The benefit of using ICE and STUN/TURN is that one can re-use the NAT traversal infrastructure already available in the Internet, such as STUN and TURN servers. Also, some middleboxes may be STUN-aware and may be able to do something "smart" when they see STUN being used for NAT traversal. However, implementing a full ICE/STUN/TURN protocol stack results in a considerable amount of effort and code which could be avoided by re-using and extending HIP messages and state machines for the same purpose. Thus, this document specifies an alternative

NAT traversal mode that uses HIP messages instead of STUN for the connectivity check keepalives and data relaying. This document also specifies how mobility management works in the context of NAT traversal, which was missing from [RFC5770].

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 [RFC2119].

This document borrows terminology from [RFC5770], [RFC7401], [RFC8046], [RFC4423], [I-D.ietf-ice-rfc5245bis], and [RFC5389]. The following terms recur in the text:

HIP relay server:
: A host that forwards any kind of HIP control packets between the Initiator and the Responder.

HIP data relay:
: A host that forwards HIP data packets, such as Encapsulating Security Payload (ESP) [RFC7402], between two hosts.

Registered host:
: A host that has registered for a relaying service with a HIP data relay.

Locator:
: As defined in [RFC8046]: "A name that controls how the packet is routed through the network and demultiplexed by the end-host. It may include a concatenation of traditional network addresses such as an IPv6 address and end-to-end identifiers such as an ESP SPI. It may also include transport port numbers or IPv6 Flow Labels as demultiplexing context, or it may simply be a network address."

LOCATOR_SET (written in capital letters):
: Denotes a HIP control packet parameter that bundles multiple locators together.

ICE offer:
: The Initiator's LOCATOR_SET parameter in a HIP I2 control packet. Corresponds to the ICE offer parameter, but is HIP specific.

ICE answer:
: The Responder's LOCATOR_SET parameter in a HIP R2 control packet. Corresponds to the ICE answer parameter, but is HIP specific.

HIP connectivity checks:
In order to obtain a non-relayed communication path, two communicating HIP hosts try to "punch holes" through their NAT boxes using this mechanism. Similar to the ICE connectivity checks, but implemented using HIP return routability checks.

Controlling host:
The controlling host nominates the candidate pair to be used with the controlled host.

Controlled host:
The controlled host waits for the controlling to nominate an address candidate pair.

Checklist:
A list of address candidate pairs that need to be tested for connectivity.

Transport address:
Transport layer port and the corresponding IPv4/v6 address.

Candidate:
A transport address that is a potential point of contact for receiving data.

Host candidate:
A candidate obtained by binding to a specific port from an IP address on the host.

Server reflexive candidate:
A translated transport address of a host as observed by a HIP relay server or a STUN/TURN server.

Peer reflexive candidate:
A translated transport address of a host as observed by its peer.

Relayed candidate:
A transport address that exists on a HIP data relay. Packets that arrive at this address are relayed towards the registered host.

Permission:
In the context of HIP data relay, permission refers to a concept similar to TURN's channels. Before a host can use a relayed candidate to forward traffic through a HIP data relay, the host must activate the relayed candidate with a specific peer host.

Base:

The base of an candidate is the local source address a host uses to send packets for the associated candidate. For example, the base of a server reflexive address is the local address the host used for registering itself to the associated HIP relay. The base of a host candidate is equal to the host candidate itself.

3. Overview of Operation

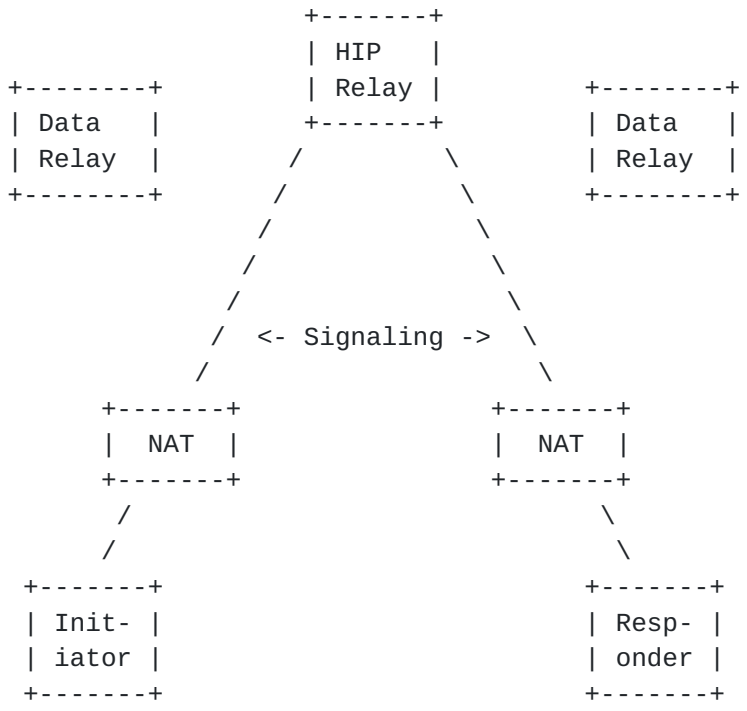


Figure 1: Example Network Configuration

In the example configuration depicted in Figure 1, both Initiator and Responder are behind one or more NATs, and both private networks are connected to the public Internet. To be contacted from behind a NAT, the Responder must be registered with a HIP relay server reachable on the public Internet, and we assume, as a starting point, that the Initiator knows both the Responder's Host Identity Tag (HIT) and the address of one of its relay servers (how the Initiator learns of the Responder's relay server is outside of the scope of this document, but may be through DNS or another name service). The Responder may have also registered to a data relay that can forward the data plane in case NAT penetration fails. It is worth noting that the HIP relay and data relay functionality may be offered by two separate servers or the same one.

The first steps are for both the Initiator and Responder to register with a relay server (need not be the same one) and gather a set of address candidates. The hosts may use HIP relay servers (or even

STUN or TURN servers) for gathering the candidates. Next, the HIP base exchange is carried out by encapsulating the HIP control packets in UDP datagrams and sending them through the Responder's relay server. As part of the base exchange, each HIP host learns of the peer's candidate addresses through the HIP offer/answer procedure embedded in the base exchange, which follows closely the ICE [I-D.ietf-ice-rfc5245bis] protocol.

Once the base exchange is completed, two HIP hosts have established a working communication session (for signaling) via a HIP relay server, but the hosts still have to find a better path, preferably without a HIP data relay, for the ESP data flow. For this, connectivity checks are carried out until a working pair of addresses is discovered. At the end of the procedure, if successful, the hosts will have established a UDP-based tunnel that traverses both NATs, with the data flowing directly from NAT to NAT or via a HIP data relay server. At this point, also the HIP signaling can be sent over the same address/port pair, and is demultiplexed from IPsec as described in the UDP encapsulation standard for IPsec [RFC3948]. Finally, the two hosts send NAT keepalives as needed in order keep their UDP-tunnel state active in the associated NAT boxes.

If either one of the hosts knows that it is not behind a NAT, hosts can negotiate during the base exchange a different mode of NAT traversal that does not use HIP connectivity checks, but only UDP encapsulation of HIP and ESP. Also, it is possible for the Initiator to simultaneously try a base exchange with and without UDP encapsulation. If a base exchange without UDP encapsulation succeeds, no HIP connectivity checks or UDP encapsulation of ESP are needed.

4. Protocol Description

This section describes the normative behavior of the protocol extension. Most of the procedures are similar to what is defined in [RFC5770] but with different, or additional, parameter types and values. In addition, a new type of relaying server, HIP data relay, is specified. Also, it should be noted that HIP version 2 [RFC7401] (instead of [RFC5201] used in [RFC5770]) is expected to be used with this NAT traversal mode.

4.1. Relay Registration

In order for two hosts to communicate over NATted environments, they need a reliable way to exchange information. HIP relay servers as defined in [RFC5770] support relaying of HIP control plane traffic over UDP in NATted environments. A HIP relay server forwards HIP control packets between the Initiator and the Responder.

To guarantee also data plane delivery over varying types of NAT devices, a host MAY also register for UDP encapsulated ESP relaying using Registration Type RELAY_UDP_ESP (value [TBD by IANA: 3]). This service may be coupled with the HIP relay server or offered separately on another server. If the server supports relaying of UDP encapsulated ESP, the host is allowed to register for a data relaying service using the registration extensions in Section 3.3 of [RFC8003]). If the server has sufficient relaying resources (free port numbers, bandwidth, etc.) available, it opens a UDP port on one of its addresses and signals the address and port to the registering host using the RELAYED_ADDRESS parameter (as defined in Section 5.12 in this document). If the relay would accept the data relaying request but does not currently have enough resources to provide data relaying service, it MUST reject the request with Failure Type "Insufficient resources" [RFC8003].

A HIP relay server MUST silently drop packets to a HIP relay client that has not previously registered with the HIP relay. The registration process follows the generic registration extensions defined in [RFC8003]. The HIP control plane relaying registration follows [RFC5770], but the data plane registration is different. It is worth noting that if the HIP control and data plane relay services reside on different hosts, the relay client has to register separately to each of them. In the example shown in Figure 2, the two services are coupled on a single host.

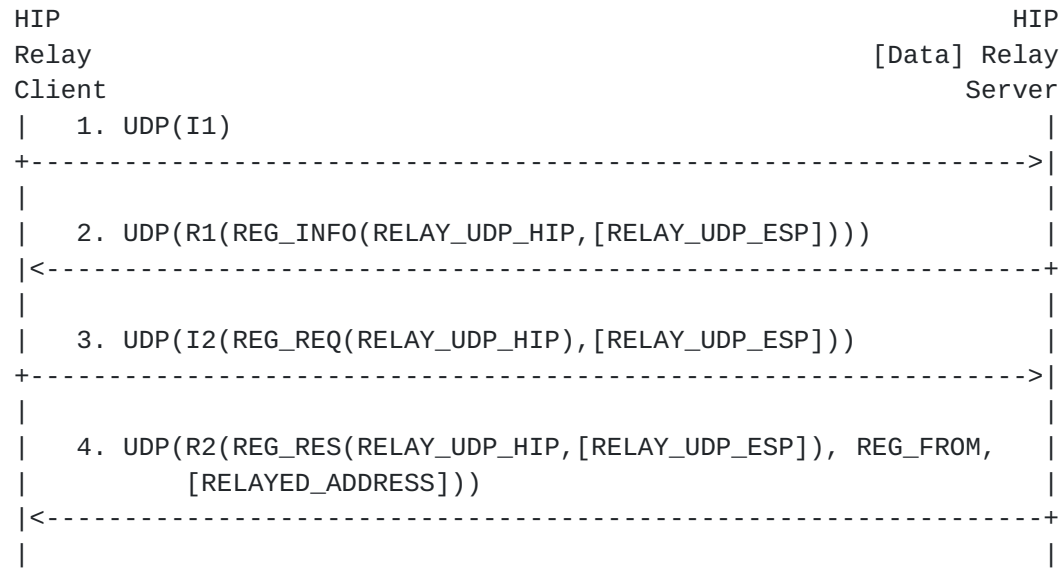


Figure 2: Example Registration with a HIP Relay

In step 1, the relay client (Initiator) starts the registration procedure by sending an I1 packet over UDP to the relay. It is

RECOMMENDED that the Initiator select a random port number from the ephemeral port range 49152-65535 for initiating a base exchange. Alternatively, a host MAY also use a single fixed port for initiating all outgoing connections. However, the allocated port MUST be maintained until all of the corresponding HIP Associations are closed. It is RECOMMENDED that the HIP relay server listen to incoming connections at UDP port 10500. If some other port number is used, it needs to be known by potential Initiators.

In step 2, the HIP relay server (Responder) lists the services that it supports in the R1 packet. The support for HIP control plane over UDP relaying is denoted by the Registration Type value RELAY_UDP_HIP (see Section 5.9). If the server supports also relaying of ESP traffic over UDP, it includes also Registration type value RELAY_UDP_ESP.

In step 3, the Initiator selects the services for which it registers and lists them in the REG_REQ parameter. The Initiator registers for HIP relay service by listing the RELAY_UDP_HIP value in the request parameter. If the Initiator requires also ESP relaying over UDP, it lists also RELAY_UDP_ESP.

In step 4, the Responder concludes the registration procedure with an R2 packet and acknowledges the registered services in the REG_RES parameter. The Responder denotes unsuccessful registrations (if any) in the REG_FAILED parameter of R2. The Responder also includes a REG_FROM parameter that contains the transport address of the client as observed by the relay (Server Reflexive candidate). If the Initiator registered to ESP relaying service, the Responder includes RELAYED_ADDRESS paramater that describes the UDP port allocated to the Initiator for ESP relaying. It is worth noting that this client must first activate this UDP port by sending an UPDATE message to the relay server that includes a PEER_PERMISSION parameter as described in Section 4.12.1 both after base exchange and handover procedures.

After the registration, the relay client sends periodically NAT keepalives to the relay server in order to keep the NAT bindings between the initiator and the relay alive. The keepalive extensions are described in Section 4.10.

The registered host MUST maintain an active HIP association with the data relay as long as it requires the data relaying service. When the HIP association is closed (or times out), or the registration lifetime passes without the registered host refreshing the registration, the data relay MUST stop relaying packets for that host and close the corresponding UDP port (unless other registered hosts are still using it).

The data relay MAY use the same relayed address and port for multiple registered hosts, but since this can cause problems with stateful firewalls (see Section 6.5) it is NOT RECOMMENDED.

When a registered client sends an UPDATE (e.g., due to host movement or to renew service registration), the relay server MUST follow the general guidelines defined in [RFC8003], with the difference that all UPDATE messages are delivered on top of UDP. In addition to this, the relay server MUST include the REG_FROM parameter in all UPDATE responses sent to the client. This applies both renewals of service registration but also to host movement, where especially the latter requires the client to learn its new server reflexive address candidate.

4.2. Transport Address Candidate Gathering

A host needs to gather a set of address candidates before contacting a non-relay host. The candidates are needed for connectivity checks that allow two hosts to discover a direct, non-relayed path for communicating with each other. One server reflexive candidate can be discovered during the registration with the HIP relay server from the REG_FROM parameter.

The candidate gathering can be done at any time, but it needs to be done before sending an I2 or R2 in the base exchange if ICE-HIP-UDP mode is to be used for the connectivity checks. It is RECOMMENDED that all three types of candidates (host, server reflexive, and relayed) are gathered to maximize the probability of successful NAT traversal. However, if no data relay is used, and the host has only a single local IP address to use, the host MAY use the local address as the only host candidate and the address from the REG_FROM parameter discovered during the relay registration as a server reflexive candidate. In this case, no further candidate gathering is needed.

If a host has more than one network interface, additional server reflexive candidates can be discovered by sending registration requests with Registration Type CANDIDATE_DISCOVERY (value [TBD by IANA: 4]) from each of the interfaces to a HIP relay server. When a HIP relay server receives a registration request with CANDIDATE_DISCOVERY type, it MUST add a REG_FROM parameter, containing the same information as if this were a relay registration, to the response. This request type SHOULD NOT create any state at the HIP relay server.

ICE guidelines for candidate gathering MUST be followed as described in section 4.1.1 in [I-D.ietf-ice-rfc5245bis]. A number of host candidates (loopback, anycast and others) should excluded as

described in section 4.1.1.1 of the ICE specification [I-D.ietf-ice-rfc5245bis]. Relayed candidates SHOULD be gathered in order to guarantee successful NAT traversal. It is RECOMMENDED for implementations to support this functionality even if it will not be used in deployments in order to enable it by software configuration update if needed at some point. A host SHOULD employ only a relay server for gathering the candidates for a single HIP association; either a one server providing both HIP and data relay functionality, or one HIP relay server and another one for data relaying if the functionality is offered by another server. When the relay service is split between two hosts, the server reflexive candidate from the HIP relay SHOULD be used instead of the one provided by the data relay. If a relayed candidate is identical to a host candidate, the relayed candidate must be discarded. NAT64 considerations in section 4.1.1.2 of [I-D.ietf-ice-rfc5245bis] apply as well.

HIP based connectivity can be utilized by IPv4 applications using LSIs and by IPv6 based applications using HITs. The LSIs and HITs of the local virtual interfaces MUST be excluded in the candidate gathering phase as well to avoid creating unnecessary loopback connectivity tests.

Gathering of candidates MAY also be performed as specified in Section 4.2 of [RFC5770] if STUN servers are available, or if the host has just a single interface and no STUN or HIP data relay servers are available.

Each local address candidate MUST be assigned a priority. The recommended formula in [I-D.ietf-ice-rfc5245bis] SHOULD be used:

```
priority = (2^24)*(type preference) + (2^8)*(local preference) +
(2^0)*(256 - component ID)
```

In the formula, type preference follows the ICE specification section 4.1.2.2 guidelines: the RECOMMENDED values are 126 for host candidates, 100 for server reflexive candidates, 110 for peer reflexive candidates, and 0 for relayed candidates. The highest value is 126 (the most preferred) and lowest is 0 (last resort). For all candidates of the same type, the preference type value MUST be identical, and, correspondingly, the value MUST be different for different types. For peer reflexive values, the type preference value MUST be higher than for server reflexive types. It should be noted that peer reflexive values are learned later during connectivity checks, so a host cannot employ it during candidate gathering stage yet.

Following the ICE specification, the local preference MUST be an integer from 0 (lowest preference) to 65535 (highest preference)

inclusive. In the case the host has only a single address candidate, the value SHOULD be 65535. In the case of multiple candidates, each local preference value MUST be unique. Dual-stack considerations for IPv6 in section 4.1.2.2 in ICE apply also here.

Unlike ICE, this protocol only creates a single UDP flow between the two communicating hosts, so only a single component exists. Hence, the component ID value MUST always be set to 1.

As defined in ICE (in section 11.3), the retransmission timeout (RTO) for address gathering from a relay SHOULD be calculated as follows:

```
RTO = MAX (500ms, Ta * (Num-Of-Pairs))
```

where Ta is the value used for Ta is the value used for the connectivity check pacing and Num-Of-Pairs is number of pairs of candidates with relay servers (e.g. in the case of a single relay server, it would be 1). A smaller value than 500 ms for the RTO MUST NOT be used.

4.3. NAT Traversal Mode Negotiation

This section describes the usage of a new non-critical parameter type. The presence of the parameter in a HIP base exchange means that the end-host supports NAT traversal extensions described in this document. As the parameter is non-critical (as defined in Section 5.2.1 of [RFC7401]), it can be ignored by an end-host, which means that the host is not required to support it or may decline to use it.

With registration with a HIP relay, it is usually sufficient to use the UDP-ENCAPSULATION mode of NAT traversal since the relay is assumed to be in public address space. Thus, the relay SHOULD propose the UDP-ENCAPSULATION mode as the preferred or only mode. The NAT traversal mode negotiation in a HIP base exchange is illustrated in Figure 3. It is worth noting that the HIP relay could be located between the hosts, but is omitted here for simplicity.

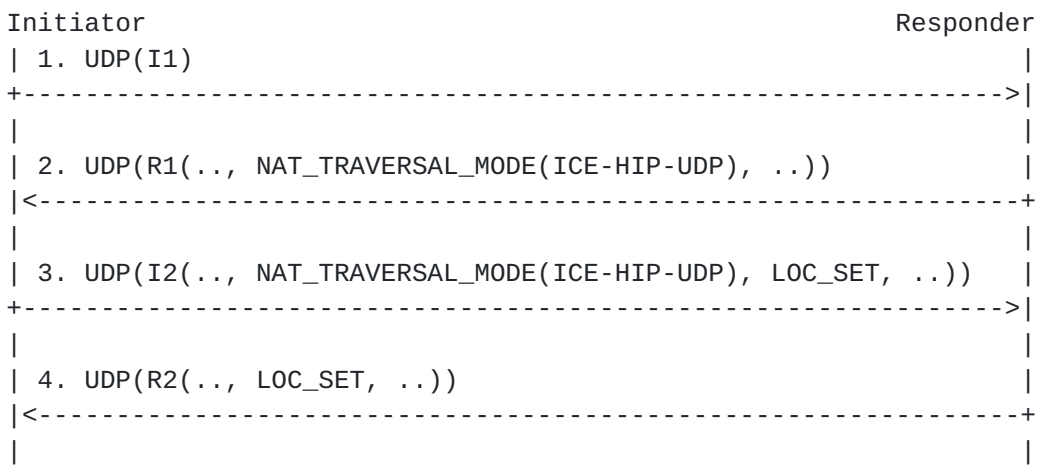


Figure 3: Negotiation of NAT Traversal Mode

In step 1, the Initiator sends an I1 to the Responder. In step 2, the Responder responds with an R1. As specified in [RFC5770], the NAT_TRAVERSAL_MODE parameter in R1 contains a list of NAT traversal modes the Responder supports. The mode specified in this document is ICE-HIP-UDP (value [TBD by IANA: 3]).

In step 3, the Initiator sends an I2 that includes a NAT_TRAVERSAL_MODE parameter. It contains the mode selected by the Initiator from the list of modes offered by the Responder. If ICE-HIP-UDP mode was selected, the I2 also includes the "Transport address" locators (as defined in Section 5.7) of the Initiator in a LOCATOR_SET parameter (denoted here LOC_SET). The locators in I2 are the "ICE offer".

In step 4, the Responder concludes the base exchange with an R2 packet. If the Initiator chose ICE NAT traversal mode, the Responder includes a LOCATOR_SET parameter in the R2 packet. The locators in R2, encoded like the locators in I2, are the "ICE answer". If the NAT traversal mode selected by the Initiator is not supported by the Responder, the Responder SHOULD reply with a NOTIFY packet with type NO_VALID_NAT_TRAVERSAL_MODE_PARAMETER and abort the base exchange.

4.4. Connectivity Check Pacing Negotiation

As explained in [RFC5770], when a NAT traversal mode with connectivity checks is used, new transactions should not be started too fast to avoid congestion and overwhelming the NATs. For this purpose, during the base exchange, hosts can negotiate a transaction pacing value, Ta, using a TRANSACTION_PACING parameter in R1 and I2 packets. The parameter contains the minimum time (expressed in milliseconds) the host would wait between two NAT traversal

transactions, such as starting a new connectivity check or retrying a previous check. The value that is used by both of the hosts is the higher of the two offered values.

The minimum Ta value SHOULD be configurable, and if no value is configured, a value of 50 ms MUST be used. Guidelines for selecting a Ta value are given in Appendix A. Hosts SHOULD NOT use values smaller than 5 ms for the minimum Ta, since such values may not work well with some NATs, as explained in [I-D.ietf-ice-rfc5245bis]. The Initiator MUST NOT propose a smaller value than what the Responder offered. If a host does not include the TRANSACTION_PACING parameter in the base exchange, a Ta value of 50 ms MUST be used as that host's minimum value.

4.5. Base Exchange via HIP Relay Server

This section describes how the Initiator and Responder perform a base exchange through a HIP relay server. Connectivity pacing (denoted as TA_P here) was described in Section 4.4 and is neither repeated here. Similarly, the NAT traversal mode negotiation process (denoted as NAT_TM in the example) was described in Section 4.3 and is neither repeated here. If a relay receives an R1 or I2 packet without the NAT traversal mode parameter, it MUST drop it and SHOULD send a NOTIFY error packet with type NO_VALID_NAT_TRAVERSAL_MODE_PARAMETER to the sender of the R1 or I2.

It is RECOMMENDED that the Initiator send an I1 packet encapsulated in UDP when it is destined to an IPv4 address of the Responder. Respectively, the Responder MUST respond to such an I1 packet with a UDP-encapsulated R1 packet, and also the rest of the communication related to the HIP association MUST also use UDP encapsulation.

Figure 4 illustrates a base exchange via a HIP relay server. We assume that the Responder (i.e. a HIP relay client) has already registered to the HIP relay server. The Initiator may have also registered to another (or the same relay server), but the base exchange will traverse always through the relay of the Responder.

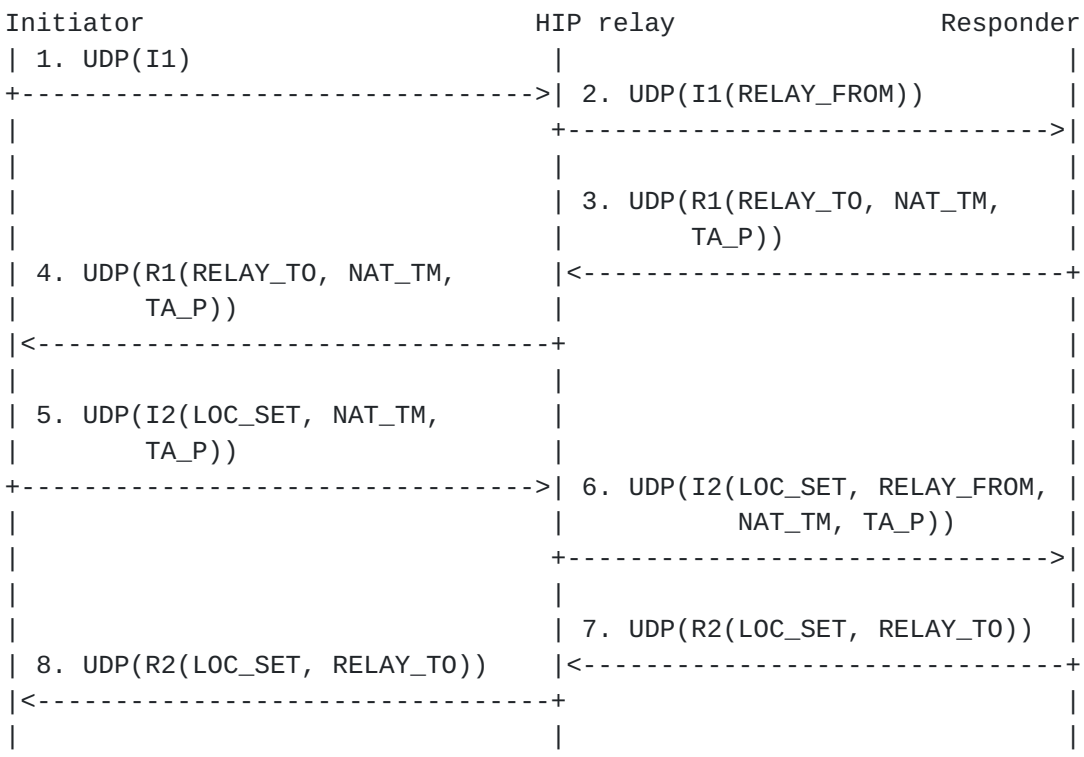


Figure 4: Base Exchange via a HIP Relay Server

In step 1 of Figure 4, the Initiator sends an I1 packet over UDP via the relay server to the Responder. In the HIP header, the source HIT belongs to the Initiator and the destination HIT to the Responder. The initiator sends the I1 packet from its IP address to the IP address of the HIP relay over UDP.

In step 2, the HIP relay server receives the I1 packet. If the destination HIT belongs to a registered Responder, the relay processes the packet. Otherwise, the relay MUST drop the packet silently. The relay appends a RELAY_FROM parameter to the I1 packet, which contains the transport source address and port of the I1 as observed by the relay. The relay protects the I1 packet with RELAY_HMAC as described in [RFC8004], except that the parameter type is different (see Section 5.8). The relay changes the source and destination ports and IP addresses of the packet to match the values the Responder used when registering to the relay, i.e., the reverse of the R2 used in the registration. The relay MUST recalculate the transport checksum and forward the packet to the Responder.

In step 3, the Responder receives the I1 packet. The Responder processes it according to the rules in [RFC7401]. In addition, the Responder validates the RELAY_HMAC according to [RFC8004] and silently drops the packet if the validation fails. The Responder

replies with an R1 packet to which it includes RELAY_TO and NAT traversal mode parameters. The responder MUST include ICE-HIP-UDP in the NAT traversal modes. The RELAY_TO parameter MUST contain the same information as the RELAY_FROM parameter, i.e., the Initiator's transport address, but the type of the parameter is different. The RELAY_TO parameter is not integrity protected by the signature of the R1 to allow pre-created R1 packets at the Responder.

In step 4, the relay receives the R1 packet. The relay drops the packet silently if the source HIT belongs to an unregistered host. The relay MAY verify the signature of the R1 packet and drop it if the signature is invalid. Otherwise, the relay rewrites the source address and port, and changes the destination address and port to match RELAY_TO information. Finally, the relay recalculates transport checksum and forwards the packet.

In step 5, the Initiator receives the R1 packet and processes it according to [RFC7401]. The Initiator MAY use the address in the RELAY_TO parameter as a local peer-reflexive candidate for this HIP association if it is different from all known local candidates. The Initiator replies with an I2 packet that uses the destination transport address of R1 as the source address and port. The I2 packet contains a LOCATOR_SET parameter that lists all the HIP candidates (ICE offer) of the Initiator. The candidates are encoded using the format defined in Section 5.7. The I2 packet MUST also contain a NAT traversal mode parameter that includes ICE-HIP-UDP mode.

In step 6, the relay receives the I2 packet. The relay appends a RELAY_FROM and a RELAY_HMAC to the I2 packet similarly as explained in step 2, and forwards the packet to the Responder.

In step 7, the Responder receives the I2 packet and processes it according to [RFC7401]. It replies with an R2 packet and includes a RELAY_TO parameter as explained in step 3. The R2 packet includes a LOCATOR_SET parameter that lists all the HIP candidates (ICE answer) of the Responder. The RELAY_TO parameter is protected by the HMAC.

In step 8, the relay processes the R2 as described in step 4. The relay forwards the packet to the Initiator. After the Initiator has received the R2 and processed it successfully, the base exchange is completed.

Hosts MUST include the address of one or more HIP relay servers (including the one that is being used for the initial signaling) in the LOCATOR_SET parameter in I2 and R2 if they intend to use such servers for relaying HIP signaling immediately after the base exchange completes. The traffic type of these addresses MUST be "HIP

signaling" and they MUST NOT be used as HIP candidates. If the HIP relay server locator used for relaying the base exchange is not included in I2 or R2 LOCATOR_SET parameters, it SHOULD NOT be used after the base exchange. Instead, further HIP signaling SHOULD use the same path as the data traffic.

4.6. Connectivity Checks

When the Initiator and Responder complete the base exchange through the HIP relay, both of them employ the IP address of the relay as the destination address for the packets. This address MUST NOT be used as a destination for ESP traffic unless the HIP relay supports also ESP data relaying. When NAT traversal mode with ICE-HIP-UDP was successfully negotiated and selected, the Initiator and Responder MUST start the connectivity checks in order to attempt to obtain direct end-to-end connectivity through NAT devices. It is worth noting that the connectivity checks MUST be completed even though no ESP_TRANSFORM would be negotiated and selected.

The connectivity checks MUST follow the ICE methodology [MMUSIC-ICE], but UDP encapsulated HIP control messages are used instead of ICE messages. Only normal nomination MUST be used for the connectivity checks, i.e., aggressive nomination MUST NOT be employed. As stated in the ICE specification, the basic procedure for connectivity checks has three phases: sorting the candidate pairs according their priority, sending checks in the prioritized order and acknowledging the checks from the peer host.

The Initiator MUST take the role of controlling host and the Responder acts as the controlled host. The roles MUST persist throughout the HIP associate lifetime (to be reused in the possibly mobility UPDATE procedures). In the case both communicating nodes are initiating the communications to each other using an I1 packet, the conflict is resolved as defined in section 6.7 in [RFC7401]: the host with the "larger" HIT changes to its Role to Responder. In such a case, the host changing its role to Responder MUST also switch to controlling role.

The protocol follows standard HIP UPDATE sending and processing rules as defined in section 6.11 and 6.12 in [RFC7401], but some new parameters are introduced (CANDIDATE_PRIORITY, MAPPED_ADDRESS and NOMINATE).

4.6.1. Connectivity Check Procedure

Figure 5 illustrates connectivity checks in a simplified scenario, where the Initiator and Responder have only a single candidate pair to check. Typically, NATs drop messages until both sides have sent

messages using the same port pair. In this scenario, the Responder sends a connectivity check first but the NAT of the Initiator drops it. However, the connectivity check from the Initiator reaches the Responder because it uses the same port pair as the first message. It is worth noting that the message flow in this section is idealistic, and, in practice, more messages would be dropped, especially in the beginning. For instance, connectivity tests always start with the candidates with the highest priority, which would be host candidates (which would not reach the recipient in this scenario).

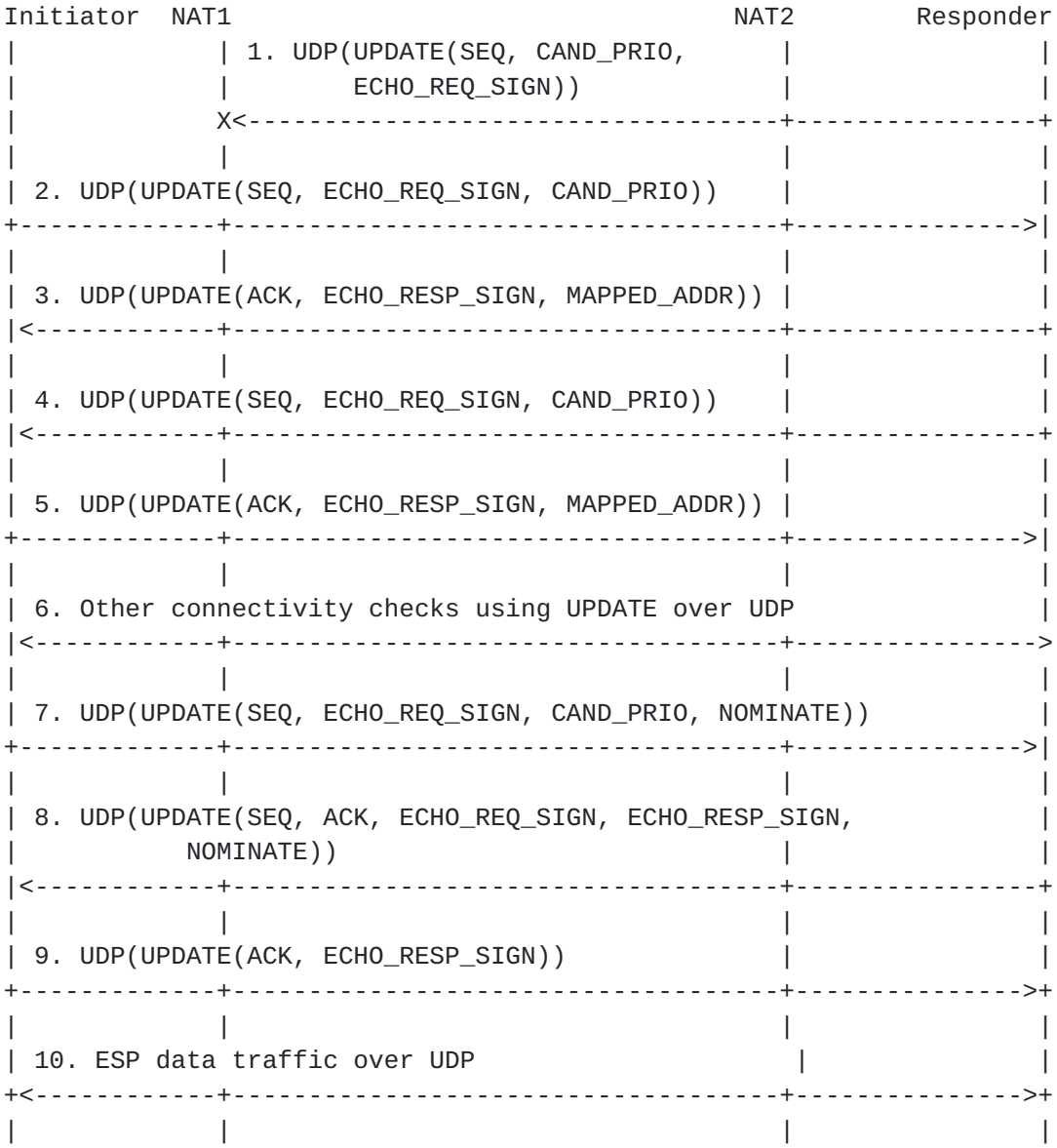


Figure 5: Connectivity Checks

In step 1, the Responder sends a connectivity check to the Initiator that the NAT of the Initiator drops. The message includes a number of parameters. As specified in [RFC7401]), the SEQ parameter includes a running sequence identifier for the connectivity check. The candidate priority (denoted "CAND_PRIO" in the figure) describes the priority of the address candidate being tested. The ECHO_REQUEST_SIGNED (denoted ECHO_REQ_SIGN in the figure) includes a nonce that the recipient must sign and echo back as it is.

In step 2, the Initiator sends a connectivity check, using the same address pair candidate as in the previous step, and the message traverses successfully the NAT boxes. The message includes the same parameters as in the previous step. It should be noted that the sequence identifier is locally assigned by the Responder, so it can be different than in the previous step.

In step 3, the Responder has successfully received the previous connectivity check from the Initiator and starts to build a response message. Since the message from the Initiator included a SEQ, the Responder must acknowledge it using an ACK parameter. Also, the nonce contained in the echo request must be echoed back in an ECHO_REQUEST_SIGNED (denoted ECHO_REQUEST_SIGN) parameter. The Responder includes also a MAPPED_ADDRESS parameter (denoted MAPPED_ADDR in the figure) that contains the transport address of the Initiator as observed by the Responder (i.e. peer reflexive candidate). This message is successfully delivered to the Initiator, and upon reception the Initiator marks the candidate pair as valid.

In step 4, the Responder retransmits the connectivity check sent in the first step, since it was not acknowledged yet.

In step 5, the Initiator responds to the previous connectivity check message from the Responder. The Initiator acknowledges the SEQ parameter from the previous message using ACK parameter and the ECHO_REQUEST_SIGN parameter with ECHO_RESPONSE_SIGNED. In addition, it includes MAPPED_ADDR parameter that includes the peer reflexive candidate. This response message is successfully delivered to the Responder, and upon reception the Initiator marks the candidate pair as valid.

In step 6, despite the two hosts now having valid address candidates, the hosts still test the remaining address candidates in a similar way as in the previous steps (due to the use of normal nomination). It should be noted that each connectivity check has a unique sequence number in the SEQ parameter.

In step 7, the Initiator has completed testing all address candidates and nominates one address candidate to be used. It sends an UPDATE

message using the selected address candidates that includes a number of parameters: SEQ, ECHO_REQUEST_SIGN, CANDIDATE_PRIORITY and the NOMINATE parameter.

In step 8, the Responder receives the message with NOMINATE parameter from the Initiator. It sends a response that includes the NOMINATE parameter in addition to a number of other parameters. The ACK and ECHO_RESPONSE_SIGNED parameters acknowledge the SEQ and ECHO_REQUEST_SIGN parameters from previous message from the Initiator. The Responder includes SEQ and ECHO_REQUEST_SIGN parameters in order to receive an acknowledgment from the Responder.

In step 9, the Initiator completes the candidate nomination process by confirming the message reception to the Responder. In the confirmation message, the ACK and ECHO_RESPONSE_SIGNED parameters correspond to the SEQ and ECHO_REQUEST_SIGN parameters in the message sent by the Responder in the previous step.

In step 10, the Initiator and Responder can start sending application payload over the successfully nominated address candidates.

It is worth noting that if either host has registered a relayed address candidate from a data relay, the host MUST activate the address before connectivity checks by sending an UPDATE message containing PEER_PERMISSION parameter as described in Section 4.12.1. Otherwise, the relay drops ESP packets using the relayed address.

4.6.2. Rules for Connectivity Checks

The HITs of the two communicating hosts MUST be used as credentials in this protocol (in contrast to ICE which employs username-password fragments). A HIT pair uniquely identifies the corresponding HIT association, and a SEQ number in an UPDATE message identifies a particular connectivity check.

All of the connectivity check packets MUST be protected with HMACs and signatures (even though the illustrations omitted them for simplicity). Each connectivity check sent by a host MUST include a SEQ parameter and ECHO_REQUEST_SIGN parameter, and correspondingly the peer MUST respond to these using ACK and ECHO_RESPONSE_SIGNED according to the rules specified in [RFC7401].

The host sending a connectivity check MUST validate that the response uses the same pair of UDP ports, and drop the packet if this is not the case.

A host may receive a connectivity check before it has received the candidates from its peer. In such a case, the host MUST immediately

generate a response, and then continue waiting for the candidates. A host MUST NOT select a candidate pair until it has verified the pair using a connectivity check as defined in section Section 4.6.1.

[RFC7401] states that UPDATE packets have to include either a SEQ or ACK parameter (or both). According to the RFC, each SEQ parameter should be acknowledged separately. In the context of NATs, this means that some of the SEQ parameters sent in connectivity checks will be lost or arrive out of order. From the viewpoint of the recipient, this is not a problem since the recipient will just "blindly" acknowledge the SEQ. However, the sender needs to be prepared for lost sequence identifiers and ACKs parameters that arrive out of order.

As specified in [RFC7401], an ACK parameter may acknowledge multiple sequence identifiers. While the examples in the previous sections do not illustrate such functionality, it is also permitted when employing ICE-HIP-UDP mode.

In ICE-HIP-UDP mode, a retransmission of a connectivity check SHOULD be sent with the same sequence identifier in the SEQ parameter. Some tested address candidates will never produce a working address pair, and thus may cause retransmissions. Upon successful nomination an address pair, a host MAY immediately stop sending such retransmissions.

ICE procedures for prioritizing candidates, eliminating redundant candidates and forming check lists (including pruning) MUST be followed as specified in [I-D.ietf-ice-rfc5245bis], with the exception that the foundation, frozen candidates and default candidates are not used. From viewpoint of the ICE specification [I-D.ietf-ice-rfc5245bis], the protocol specified in this document operates using Component ID of 1 on all candidates, and the foundation of all candidates is unique. This specification defines only "full ICE" mode, and the "lite ICE" is not supported. The reasoning behind the missing features is described in Appendix B.

The connectivity check messages MUST be paced by the Ta value negotiated during the base exchange as described in Section 4.4. If neither one of the hosts announced a minimum pacing value, a value of 20 ms SHOULD be used.

Both hosts MUST form a priority ordered checklist and begin to check transactions every Ta milliseconds as long as the checks are running and there are candidate pairs whose tests have not started. The retransmission timeout (RTO) for the connectivity check UPDATE packets SHOULD be calculated as follows:

RTO = MAX (500ms, Ta * (Num-Waiting + Num-In-Progress))

In the RTO formula, Ta is the value used for the connectivity check pacing, Num-Waiting is the number of pairs in the checklist in the "Waiting" state, and Num-In-Progress is the number of pairs in the "In-Progress" state. This is identical to the formula in [I-D.ietf-ice-rfc5245bis] when there is only one checklist. A smaller value than 500 ms for the RTO MUST NOT be used.

Each connectivity check request packet MUST contain a CANDIDATE_PRIORITY parameter (see Section 5.14) with the priority value that would be assigned to a peer reflexive candidate if one was learned from the corresponding check. An UPDATE packet that acknowledges a connectivity check request MUST be sent from the same address that received the check and delivered to the same address where the check was received from. Each acknowledgment UPDATE packet MUST contain a MAPPED_ADDRESS parameter with the port, protocol, and IP address of the address where the connectivity check request was received from.

Following ICE guidelines [I-D.ietf-ice-rfc5245bis], it is RECOMMENDED to restrict the total number of connectivity checks to 100 for each host association. This can be achieved by limiting the connectivity checks to the 100 candidate pairs with the highest priority.

4.6.3. Rules for Concluding Connectivity Checks

The controlling agent may find multiple working candidate pairs. To conclude the connectivity checks, it SHOULD nominate the pair with the highest priority. The controlling agent MUST nominate a candidate pair essentially by repeating a connectivity check using an UPDATE message that contains a SEQ parameter (with new sequence number), a ECHO_REQUEST_SIGN parameter, the priority of the candidate in a CANDIDATE_PRIORITY parameter and NOMINATE parameter to signify conclusion of the connectivity checks. Since the nominated address pair has already been tested for reachability, the controlled host should be able to receive the message. Upon reception, the controlled host SHOULD select the nominated address pair. The response message MUST include a SEQ parameter with a new sequence id, acknowledgment of the sequence from the controlling host in an ACK parameter, a new ECHO_REQUEST_SIGNED parameter, ECHO_RESPONSE_SIGNED parameter corresponding to the ECHO_REQUEST_SIGNED parameter from the controlling host and the NOMINATE parameter. After sending this packet, the controlled host can create IPsec security associations using the nominated address candidate for delivering application payload to the controlling host. Since the message from the controlled host included a new sequence id and echo request for signature, the controlling host MUST acknowledge this with a new

UPDATE message that includes an ACK and ECHO_RESPONSE_SIGNED parameters. After this final concluding message, the controlling host also can create IPsec security associations for delivering application payload to the controlled host.

It is possible that packets are delayed by the network. Both hosts MUST continue to respond to any connectivity checks despite an address pair having been nominated.

If all the connectivity checks have failed, the hosts MUST NOT send ESP traffic to each other but MAY continue communicating using HIP packets and the locators used for the base exchange. Also, the hosts SHOULD notify each other about the failure with a CONNECTIVITY_CHECKS_FAILED NOTIFY packet (see Section 5.10).

4.7. NAT Traversal Alternatives

4.7.1. Minimal NAT Traversal Support

If the Responder has a fixed and publicly reachable IPv4 address and does not employ a HIP relay, the explicit NAT traversal mode negotiation MAY be omitted, and thus even the UDP-ENCAPSULATION mode does not have to be negotiated. In such a scenario, the Initiator sends an I1 message over UDP and the Responder responds with an R1 message over UDP without including any NAT traversal mode parameter. The rest of the base exchange follows the procedures defined in [RFC7401], except that the control and data plane use UDP encapsulation. Here, the use of UDP for NAT traversal is agreed implicitly. This way of operation is still subject to NAT timeouts, and the hosts MUST employ NAT keepalives as defined in Section 4.10.

4.7.2. Base Exchange without Connectivity Checks

It is possible to run a base exchange without any connectivity checks as defined in section 4.8 in [RFC5770]. The procedure is applicable also in the context of this specification, so it is repeated here for completeness.

In certain network environments, the connectivity checks can be omitted to reduce initial connection set-up latency because a base exchange acts as an implicit connectivity test itself. For this to work, the Initiator MUST be able to reach the Responder by simply UDP encapsulating HIP and ESP packets sent to the Responder's address. Detecting and configuring this particular scenario is prone to failure unless carefully planned.

In such a scenario, the Responder MAY include UDP-ENCAPSULATION NAT traversal mode as one of the supported modes in the R1 packet. If

the Responder has registered to a HIP relay server, it MUST also include a LOCATOR_SET parameter in R1 that contains a preferred address where the Responder is able to receive UDP-encapsulated ESP and HIP packets. This locator MUST be of type "Transport address", its Traffic type MUST be "both", and it MUST have the "Preferred bit" set (see Table 1). If there is no such locator in R1, the source address of R1 is used as the Responder's preferred address.

The Initiator MAY choose the UDP-ENCAPSULATION mode if the Responder listed it in the supported modes and the Initiator does not wish to use the connectivity checks defined in this document for searching for a more optimal path. In this case, the Initiator sends the I2 with UDP-ENCAPSULATION mode in the NAT traversal mode parameter directly to the Responder's preferred address (i.e., to the preferred locator in R1 or to the address where R1 was received from if there was no preferred locator in R1). The Initiator MAY include locators in I2 but they MUST NOT be taken as address candidates, since connectivity checks defined in this document will not be used for connections with UDP-ENCAPSULATION NAT traversal mode. Instead, if R2 and I2 are received and processed successfully, a security association can be created and UDP-encapsulated ESP can be exchanged between the hosts after the base exchange completes. However, the Responder SHOULD NOT send any ESP to the Initiator's address before it has received data from the Initiator, as specified in Sections 4.4.3. and 6.9 of [RFC7401] and in Sections 3.2.9 and 5.4 of [RFC8046].

Since an I2 packet with UDP-ENCAPSULATION NAT traversal mode selected MUST NOT be sent via a relay, the Responder SHOULD reject such I2 packets and reply with a NO_VALID_NAT_TRAVERSAL_MODE_PARAMETER NOTIFY packet (see Section 5.10).

If there is no answer for the I2 packet sent directly to the Responder's preferred address, the Initiator MAY send another I2 via the HIP relay server, but it MUST NOT choose UDP-ENCAPSULATION NAT traversal mode for that I2.

4.7.3. Initiating a Base Exchange both with and without UDP Encapsulation

It is possible to run a base exchange in parallel both with and without UDP encapsulation as defined in section 4.9 in [RFC5770]. The procedure is applicable also in the context of this specification, so it is repeated here for completeness.

The Initiator MAY also try to simultaneously perform a base exchange with the Responder without UDP encapsulation. In such a case, the Initiator sends two I1 packets, one without and one with UDP

encapsulation, to the Responder. The Initiator MAY wait for a while before sending the other I1. How long to wait and in which order to send the I1 packets can be decided based on local policy. For retransmissions, the procedure is repeated.

The I1 packet without UDP encapsulation may arrive directly, without any relays, at the Responder. When this happens, the procedures in [RFC7401] are followed for the rest of the base exchange. The Initiator may receive multiple R1 packets, with and without UDP encapsulation, from the Responder. However, after receiving a valid R1 and answering it with an I2, further R1 packets that are not retransmissions of the original R1 message MUST be ignored.

The I1 packet without UDP encapsulation may also arrive at a HIP-capable middlebox. When the middlebox is a HIP rendezvous server and the Responder has successfully registered with the rendezvous service, the middlebox follows rendezvous procedures in [RFC8004].

If the Initiator receives a NAT traversal mode parameter in R1 without UDP encapsulation, the Initiator MAY ignore this parameter and send an I2 without UDP encapsulation and without any selected NAT traversal mode. When the Responder receives the I2 without UDP encapsulation and without NAT traversal mode, it will assume that no NAT traversal mechanism is needed. The packet processing will be done as described in [RFC7401]. The Initiator MAY store the NAT traversal modes for future use, e.g., in case of a mobility or multihoming event that causes NAT traversal to be used during the lifetime of the HIP association.

4.8. Sending Control Packets after the Base Exchange

The same considerations of sending control packets after the base exchange described in section 5.10 in [RFC5770] apply also here, so they are repeated here for completeness.

After the base exchange, the end-hosts MAY send HIP control packets directly to each other using the transport address pair established for a data channel without sending the control packets through the HIP relay server. When a host does not get acknowledgments, e.g., to an UPDATE or CLOSE packet after a timeout based on local policies, the host SHOULD resend the packet through the relay, if it was listed in the LOCATOR_SET parameter in the base exchange.

If control packets are sent through a HIP relay server, the host registered with the relay MUST utilize the RELAY_TO parameter as in the base exchange. The HIP relay server SHOULD forward HIP packets to the registered hosts and forward packets from a registered host to the address in the RELAY_TO parameter. The relay MUST add a

RELAY_FROM parameter to the control packets it relays to the registered hosts.

If the HIP relay server is not willing or able to relay a HIP packet, it MAY notify the sender of the packet with MESSAGE_NOT_RELAYED error notification (see Section 5.10).

4.9. Mobility Handover Procedure

A host may move after base exchange and connectivity checks. Mobility extensions for HIP [RFC8046] define handover procedures without NATs. In this section, we define how two hosts interact with handover procedures in scenarios involving NATs. The specified extensions define only simple mobility using a pair of security associations, and multihoming extensions are left to be defined in later specifications. The procedures in this section offer the same functionality as "ICE restart" specified in [I-D.ietf-ice-rfc5245bis]. The example described in this section shows only a relay server for the peer host for the sake of simplicity, but also the mobile host may also have a relay server.

The assumption here is that the two hosts have successfully negotiated and chosen the ICE-HIP-UDP mode during the base exchange as defined in Section 4.3. The Initiator of the base exchange MUST store information that it was the controlling host during the base exchange. Similarly, the Responder MUST store information that it was the controlled host during the base exchange.

Prior to starting the handover procedures with all peer hosts, the mobile host SHOULD first send UPDATE messages to its HIP and data relays if it has registered to such. It SHOULD wait for all of them to respond for two minutes and then continue with the handover procedure without information from the relays that did not respond. As defined in section Section 4.1, a response message from a relay includes a REG_FROM parameter that describes the server reflexive candidate of the mobile host to be used in the candidate exchange during the handover. Similarly, an UPDATE to a data relay is necessary to make sure the data relay can forward data to the correct IP address after a handoff.

The mobility extensions for NAT traversal are illustrated in Figure 6. The mobile host is the host that has changed its locators, and the peer host is the host it has a host association with. The mobile host may have multiple peers and it repeats the process with all of its peers. In the figure, the HIP relay belongs to the peer host, i.e., the peer host is a relay client for the HIP relay. Next, we describe the procedure in the figure in detail.

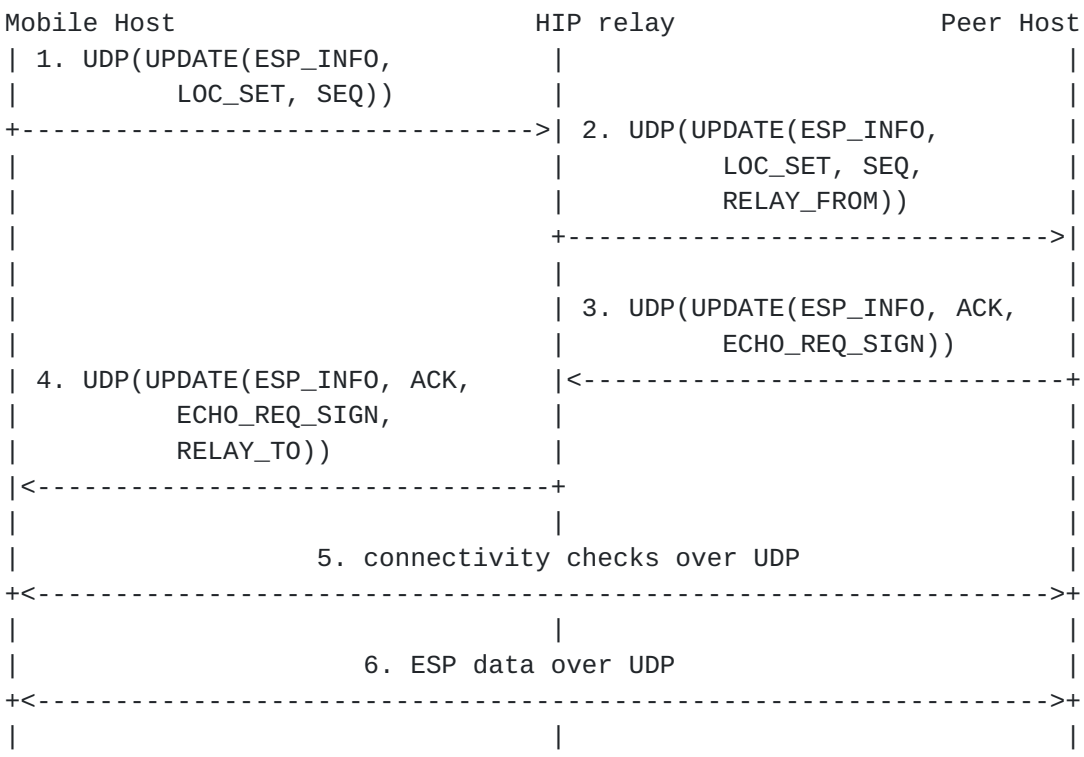


Figure 6: HIP UPDATE procedure

In step 1, the mobile host has changed location and sends a location update to its peer through the HIP relay of the peer. It sends an UPDATE packet with source HIT belonging to itself and destination HIT belonging to the peer host. In the packet, the source IP address belongs to the mobile host and the destination to the HIP relay. The packet contains an ESP_INFO parameter, where, in this case, the OLD SPI and NEW SPI parameters both contain the pre-existing incoming SPI. The packet also contains the locators of the mobile host in a LOCATOR_SET parameter. The packet contains also a SEQ number to be acknowledged by the peer. As specified in [RFC8046], the packet may also include a HOST_ID (for middlebox inspection) and DIFFIE_HELLMAN parameter for rekeying.

In step 2, the HIP relay receives the UPDATE packet and forwards it to the peer host (i.e. relay client). The HIP relay rewrites the destination IP address and appends a RELAY_FROM parameter to the message.

In step 3, the peer host receives the UPDATE packet, processes it and responds with another UPDATE message. The message is destined to the HIT of mobile host and to the IP address of the HIP relay. The message includes an ESP_INFO parameter where, in this case, the OLD SPI and NEW SPI parameters both contain the pre-existing incoming

SPI. The peer includes a new SEQ and ECHO_REQUEST_SIGNED parameters to be acknowledged by the mobile host. The message acknowledges the SEQ parameter of the earlier message with an ACK parameter. After this step, the peer host can initiate the connectivity checks.

In step 4, the HIP relay receives the message, rewrites the destination IP address, appends an RELAY_TO parameter and forwards the modified message to the mobile host. When mobile host has processed the message successfully, it can initiate the connectivity checks.

In step 5, the two hosts test for connectivity across NATs according to procedures described in Section 4.6. The original Initiator of the communications is the controlling and the original Responder is the controlled host.

In step 6, the connectivity checks are successfully completed and the controlling host has nominated one address pair to be used. The hosts set up security associations to deliver the application payload.

If either host has registered a relayed address candidate from a data relay, the host MUST reactivate the address before connectivity checks by sending an UPDATE message containing PEER_PERMISSION parameter as described in Section 4.12.1. Otherwise, the relay drops ESP packets using the relayed address.

4.10. NAT Keepalives

To prevent NAT states from expiring, communicating hosts send periodic keepalives to other hosts with which they have established a host associating. Both a registered client and relay server SHOULD send HIP NOTIFY packets to each other every 15 seconds (the so called Tr value in ICE) unless they have exchange some other traffic over the used UDP ports. Other values MAY be used, but a Tr value smaller than 15 seconds MUST NOT be used. The keepalive message encoding format is defined in Section 5.3. If the base exchange or mobility handover procedure occurs during an extremely slow path, a host MAY also send HIP NOTIFY packet every 15 seconds to keep the path active with the recipient.

4.11. Closing Procedure

The two-way procedure for closing a HIP association and the related security associations is defined in [RFC7401]. One host initiates the procedure by sending a CLOSE message and the recipient confirms it with CLOSE_ACK. All packets are protected using HMACs and

signatures, and the CLOSE messages includes a ECHO_REQUEST_SIGNED parameter to protect against replay attacks.

The same procedure for closing HIP associations applies also here, but the messaging occurs using the UDP encapsulated tunnel that the two hosts employ. A host sending the CLOSE message SHOULD first send the message over a direct link. After a number of retransmissions, it MUST send over a HIP relay of the recipient if one exists. The host receiving the CLOSE message directly without a relay SHOULD respond directly. If CLOSE message came via a relay, the host SHOULD respond using the same relay.

4.12. Relaying Considerations

4.12.1. Forwarding Rules and Permissions

The HIP data relay uses a similar permission model as a TURN server: before the data relay forwards any ESP data packets from a peer to a registered host (or the other direction), the client MUST set a permission for the peer's address. The permissions also install a forwarding rule for each direction, similar to TURN's channels, based on the Security Parameter Index (SPI) values in the ESP packets.

Permissions are not required for HIP control packets. However, if a relayed address (as conveyed in the RELAYED_ADDRESS parameter from the data relay) is selected to be used for data, the registered host MUST send an UPDATE message to the data relay containing a PEER_PERMISSION parameter (see Section 5.13) with the address of the peer, and the outbound and inbound SPI values the registered host is using with this particular peer. To avoid packet dropping of ESP packets, the registered host SHOULD send the PEER_PERMISSION parameter before connectivity checks both in the case of base exchange and a mobility handover. It is worth noting that the UPDATE message includes a SEQ parameter (as specified in [RFC7401]) that the data relay must acknowledge, so that the registered host can resend the message with PEER_PERMISSION parameter if it gets lost.

When a data relay receives an UPDATE with a PEER_PERMISSION parameter, it MUST check if the sender of the UPDATE is registered for data relaying service, and drop the UPDATE if the host was not registered. If the host was registered, the relay checks if there is a permission with matching information (address, protocol, port and SPI values). If there is no such permission, a new permission MUST be created and its lifetime MUST be set to 5 minutes. If an identical permission already existed, it MUST be refreshed by setting the lifetime to 5 minutes. A registered host SHOULD refresh permissions 1 minute before the expiration when the permission is still needed.

When a data relay receives an UPDATE from a registered client but without a PEER_PERMISSION parameter and with a new locator set, the relay can assume that the mobile host has changed its location and, thus, is not reachable in its previous location. In such an event, the data relay SHOULD deactivate the permission and stop relaying data plane traffic to the client.

The relayed address MUST be activated with the PEER_PERMISSION parameter both after a base exchange and after a handover procedure with another ICE-HIP-UDP capable host. Unless activated, the data relay MUST drop all ESP packets. It is worth noting that a relay client does not have to renew its registration upon a change of location UPDATE, but only when the lifetime of the registration is close to end.

4.12.2. HIP Data Relay and Relaying of Control Packets

When a HIP data relay accepts to relay UDP encapsulated ESP between a registered host and its peer, the relay opens a UDP port (relayed address) for this purpose as described in Section 4.1. This port can be used for delivering also control packets because connectivity checks also cover the path through the data relay. If the data relay receives a UDP encapsulated HIP control packet on that port, it MUST forward the packet to the registered host and add a RELAY_FROM parameter to the packet as if the data relay were acting as a HIP relay server. When the registered host replies to a control packet with a RELAY_FROM parameter via its relay, the registered host MUST add a RELAY_TO parameter containing the peer's address and use the address of its data relay as the destination address. Further, the data relay MUST send this packet to the peer's address from the relayed address.

If the data relay receives a UDP packet that is not a HIP control packet to the relayed address, it MUST check if it has a permission set for the peer the packet is arriving from (i.e., the sender's address and SPI value matches to an installed permission). If permissions are set, the data relay MUST forward the packet to the registered host that created the permission. The data relay MUST also implement the similar checks for the reverse direction (i.e. ESP packets from the registered host to the peer). Packets without a permission MUST be dropped silently.

4.12.3. Handling Conflicting SPI Values

The inbound SPI values of the registered clients should be unique so that a data relay can properly demultiplex incoming packets from peer hosts to the correct registered clients. Likewise, the inbound SPIs

of the peer hosts should be unique for the same reason. These two cases are discussed in this section separately.

In first case, the SPI collision problem occurs when two Initiators run a base exchange to the same Responder (i.e. registered host), and both the Initiators claim the same inbound SPI. This is not a problem for Responder since the two Initiators can be distinguished by their transport addresses. However, it is an issue for the data relay because the it cannot demultiplex packets from the Initiator to the correct Responder. Thus, upon receiving an I2 with a colliding SPI, the Responder MUST NOT include the relayed address candidate in the R2 message because the data relay would not be able demultiplex the related ESP packet to the correct Initiator. The same applies also the handover procedure; the registered host MUST NOT include the relayed address candidate when sending its new locator set in an UPDATE to its peer if it would cause a SPI conflict with another peer. Since the SPI space is 32 bits and the SPI values should be random, the probability for a conflicting SPI value is fairly small. However, a registered host with many peers MAY proactively decrease the odds of a conflict by registering to multiple data relays. The described collision scenario can be avoided if the Responder delivers a new relayed address candidate upon SPI collisions. Each relayed address has a separate UDP port reserved to it, so the relay can demultiplex properly conflicting SPIs of the Initiators based on the SPI and port number towards the correct Responder.

In the second case, the SPI collision problems occurs if two hosts have registered to the same data relay and a third host initiates base exchange with both of them. In this case, the data relay has allocated separate UDP ports for the two registered hosts acting now as Responders. When the Responders send identical SPI values in their I2 messages via the relay, the relay can properly deliver the message to the correct Responder because the UDP ports are different.

5. Packet Formats

The following subsections define the parameter and packet encodings for the HIP and ESP packets. All values MUST be in network byte order.

It is worth noting that most of the parameters are shown for the sake of completeness even though they are specified already in [RFC5770]. New parameters are explicitly described as new.

5.1. HIP Control Packets

Figure 7 illustrates the packet format for UDP-encapsulated HIP. The format is identical to [RFC5770].

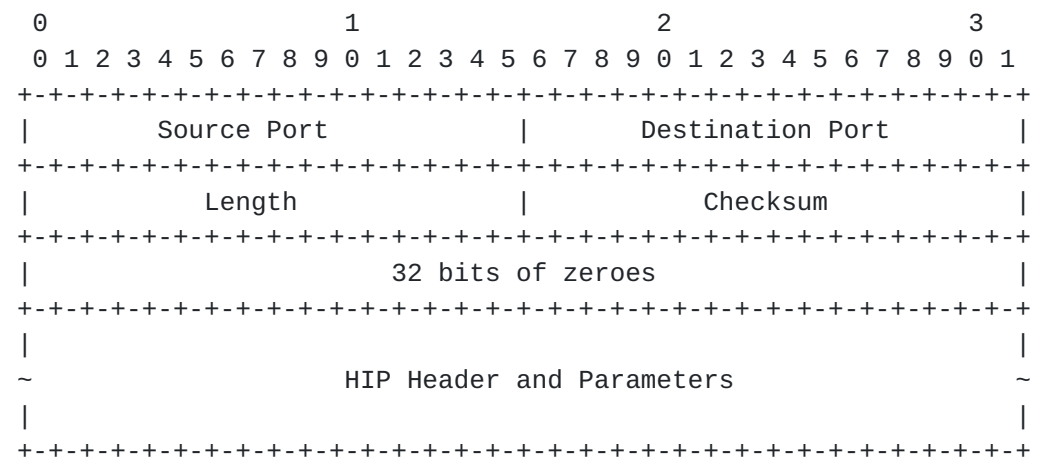


Figure 7: Format of UDP-Encapsulated HIP Control Packets

HIP control packets are encapsulated in UDP packets as defined in Section 2.2 of [RFC3948], "IKE Header Format for Port 4500", except that a different port number is used. Figure 7 illustrates the encapsulation. The UDP header is followed by 32 zero bits that can be used to differentiate HIP control packets from ESP packets. The HIP header and parameters follow the conventions of [RFC7401] with the exception that the HIP header checksum MUST be zero. The HIP header checksum is zero for two reasons. First, the UDP header already contains a checksum. Second, the checksum definition in [RFC7401] includes the IP addresses in the checksum calculation. The NATs that are unaware of HIP cannot recompute the HIP checksum after changing IP addresses.

A HIP relay server or a Responder without a relay SHOULD listen at UDP port 10500 for incoming UDP-encapsulated HIP control packets. If some other port number is used, it needs to be known by potential Initiators.

5.2. Connectivity Checks

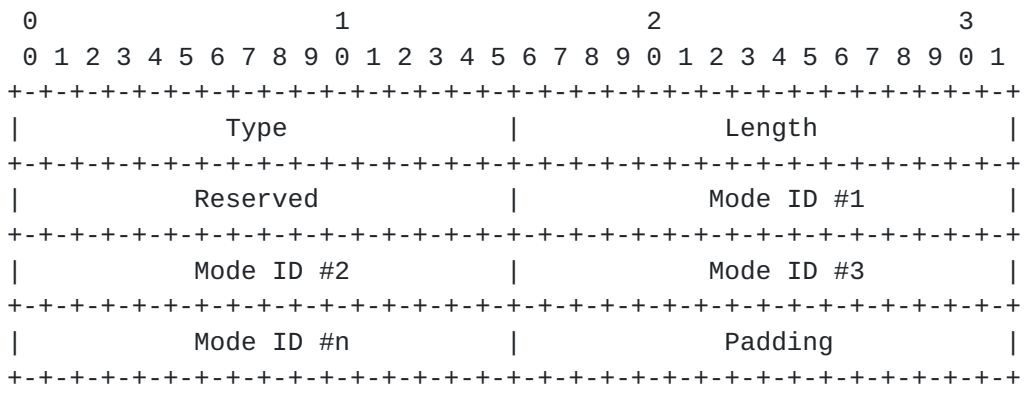
HIP connectivity checks are HIP UPDATE packets. The format is specified in [RFC7401].

5.3. Keepalives

The RECOMMENDED encoding format for keepalives is HIP NOTIFY packets as specified in [RFC7401] with Notify message type field set to NAT_KEEPALIVE [TBD by IANA: 16384] and with an empty Notification data field. It is worth noting that sending of such a HIP NOTIFY message MAY be omitted if the host is actively (or passively) sending other traffic to the peer host over the UDP tunnel associate with the host association (and IPsec security associations since the same port pair is reused) during the Tr period. For instance, the host MAY actively send ICMPv6 requests (or respond with an ICMPv6 response) inside the ESP tunnel to test the health of the associated IPsec security associations. Alternatively, the host MAY use UPDATE packets as a substitute. A minimal UPDATE packet would consist of a SEQ and ECHO_REQ_SIGN parameters, and a more complex would involve rekeying procedures as specified in section 6.8 in [RFC7402]. It is worth noting that a host actively sending periodic UPDATE packets to a busy server may increase the computational load of the server since it has to verify HMACs and signatures in UPDATE messages.

5.4. NAT Traversal Mode Parameter

The format of NAT traversal mode parameter is borrowed from [RFC5770]. The format of the NAT_TRAVERSAL_MODE parameter is similar to the format of the ESP_TRANSFORM parameter in [RFC7402] and is shown in Figure 8. This specification defines the traversal mode identifier for ICE-HIP-UDP and reuses the UDP-ENCAPSULATION mode from [RFC5770]. The identifier named RESERVED is reserved for future use. Future specifications may define more traversal modes.



Type 608
Length length in octets, excluding Type, Length, and padding
Reserved zero when sent, ignored when received
Mode ID defines the proposed or selected NAT traversal mode(s)

The following NAT traversal mode IDs are defined:

ID name	Value
RESERVED	0
UDP-ENCAPSULATION	1
ICE-HIP-UDP	3

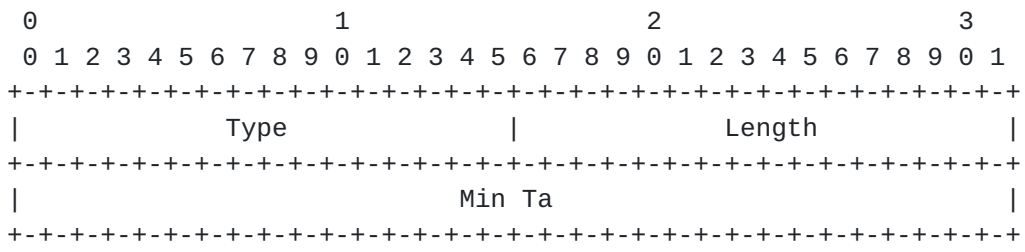
Figure 8: Format of the NAT_TRAVERSAL_MODE Parameter

The sender of a NAT_TRAVERSAL_MODE parameter MUST make sure that there are no more than six (6) Mode IDs in one NAT_TRAVERSAL_MODE parameter. Conversely, a recipient MUST be prepared to handle received NAT traversal mode parameters that contain more than six Mode IDs by accepting the first six Mode IDs and dropping the rest. The limited number of Mode IDs sets the maximum size of the NAT_TRAVERSAL_MODE parameter. The modes MUST be in preference order, most preferred mode(s) first.

Implementations conforming to this specification MUST implement UDP-ENCAPSULATION and SHOULD implement ICE-HIP-UDP modes.

5.5. Connectivity Check Transaction Pacing Parameter

The TRANSACTION_PACING is a new parameter, and it shown in Figure 9 contains only the connectivity check pacing value, expressed in milliseconds, as a 32-bit unsigned integer.

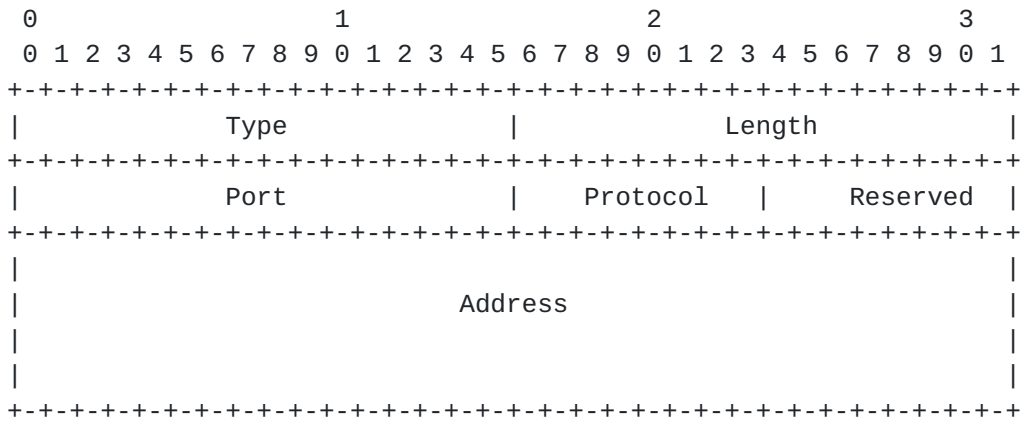


Type 610
Length 4
Min Ta the minimum connectivity check transaction pacing value the host would use (in milliseconds)

Figure 9: Format of the TRANSACTION_PACING Parameter

5.6. Relay and Registration Parameters

The format of the REG_FROM, RELAY_FROM, and RELAY_TO parameters is shown in Figure 10. All parameters are identical except for the type. REG_FROM is the only parameter covered with the signature.



Type REG_FROM: 950
RELAY_FROM: 63998
RELAY_TO: 64002
Length 20
Port transport port number; zero when plain IP is used
Protocol IANA assigned, Internet Protocol number. 17 for UDP, 0 for plain IP
Reserved reserved for future use; zero when sent, ignored when received
Address an IPv6 address or an IPv4 address in "IPv4-Mapped IPv6 address" format

Figure 10: Format of the REG_FROM, RELAY_FROM, and RELAY_TO Parameters

REG_FROM contains the transport address and protocol from which the HIP relay server sees the registration coming. RELAY_FROM contains the address from which the relayed packet was received by the relay server and the protocol that was used. RELAY_TO contains the same information about the address to which a packet should be forwarded.

5.7. LOCATOR_SET Parameter

This specification reuses the format for UDP-based locators specified in [RFC5770] to be used for communicating the address candidates between two hosts. The generic and NAT-traversal-specific locator parameters are illustrated in Figure 11.

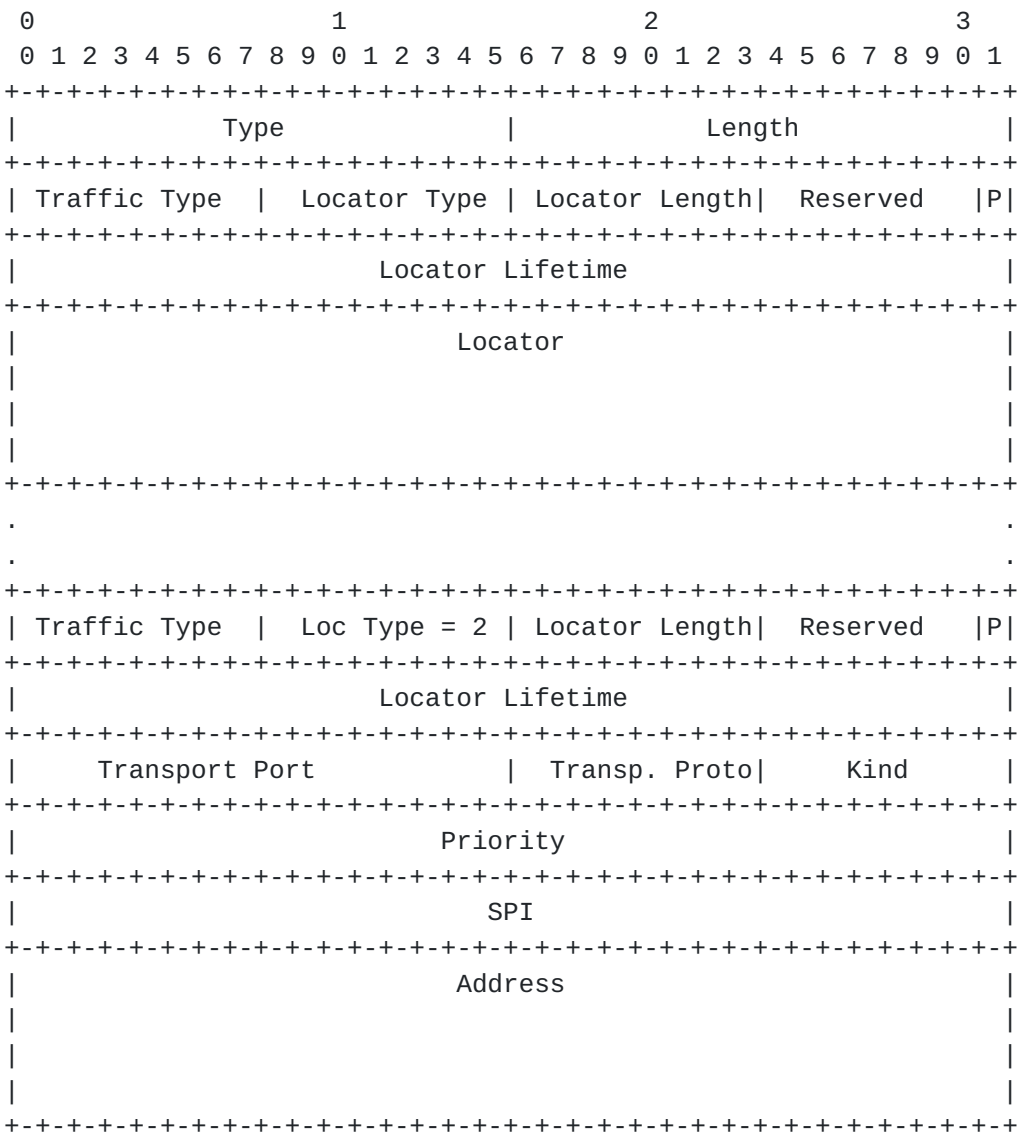


Figure 11: LOCATOR_SET Parameter

The individual fields in the LOCATOR_SET parameter are described in Table 1.

Field	Value(s)	Purpose
Type	193	Parameter type
Length	Variable	Length in octets, excluding Type and Length fields and padding
Traffic Type	0-2	Is the locator for HIP signaling (1), for ESP (2), or for both (0)
Locator Type	2	"Transport address" locator type
Locator Length	7	Length of the fields after Locator
Reserved	0	Reserved for future extensions
Preferred (P) bit	0 or 1	Set to 1 for a Locator in R1 if the Responder can use it for the rest of the base exchange, otherwise set to zero
Locator Lifetime	Variable	Locator lifetime in seconds
Transport Port	Variable	Transport layer port number
Transport Protocol	Variable	IANA assigned, transport layer Internet Protocol number. Currently only UDP (17) is supported.
Kind	Variable	0 for host, 1 for server reflexive, 2 for peer reflexive or 3 for relayed address
Priority	Variable	Locator's priority as described in [I-D.ietf-ice-rfc5245bis]. It is worth noting that while the priority of a single locator candidate is 32-bits, but an implementation should use a 64-bit integer to calculate the priority of a candidate pair for the ICE priority algorithm.
SPI	Variable	Security Parameter Index (SPI) value that the host expects to see in incoming ESP packets that use this locator
Address	Variable	IPv6 address or an "IPv4-Mapped IPv6 address" format IPv4 address [RFC4291]

Table 1: Fields of the LOCATOR_SET Parameter

5.8. RELAY_HMAC Parameter

As specified in [RFC5770], the RELAY_HMAC parameter value has the TLV type 65520. It has the same semantics as RVS_HMAC [RFC8004].

5.9. Registration Types

The REG_INFO, REG_REQ, REG_RESP, and REG_FAILED parameters contain Registration Type [RFC8003] values for HIP relay server registration. The value for RELAY_UDP_HIP is 2 as specified in [RFC5770].

5.10. Notify Packet Types

A HIP relay server and end-hosts can use NOTIFY packets to signal different error conditions. The NOTIFY packet types are the same as in [RFC5770].

The Notify Packet Types [RFC7401] are shown below. The Notification Data field for the error notifications SHOULD contain the HIP header of the rejected packet and SHOULD be empty for the CONNECTIVITY_CHECKS_FAILED type.

NOTIFICATION PARAMETER - ERROR TYPES	Value
NO_VALID_NAT_TRAVERSAL_MODE_PARAMETER	60

If a HIP relay server does not forward a base exchange packet due to missing NAT traversal mode parameter, or the Initiator selects a NAT traversal mode that the Responder did not expect, the relay or the Responder may send back a NOTIFY error packet with this type.

CONNECTIVITY_CHECKS_FAILED	61

Used by the end-hosts to signal that NAT traversal connectivity checks failed and did not produce a working path.

MESSAGE_NOT_RELAYED	62

Used by a HIP relay server to signal that is was not able or willing to relay a HIP packet.

5.11. ESP Data Packets

The format for ESP data packets is identical to [RFC5770].

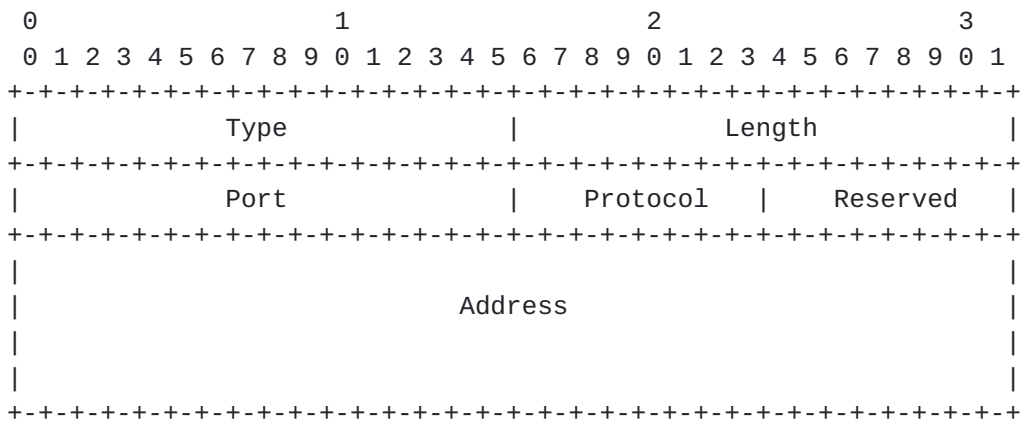
[RFC3948] describes the UDP encapsulation of the IPsec ESP transport and tunnel mode. On the wire, the HIP ESP packets do not differ from the transport mode ESP, and thus the encapsulation of the HIP ESP packets is same as the UDP encapsulation transport mode ESP. However, the (semantic) difference to Bound End-to-End Tunnel (BEET) mode ESP packets used by HIP is that IP header is not used in BEET integrity protection calculation.

During the HIP base exchange, the two peers exchange parameters that enable them to define a pair of IPsec ESP security associations (SAs) as described in [RFC7402]. When two peers perform a UDP-encapsulated base exchange, they MUST define a pair of IPsec SAs that produces UDP-encapsulated ESP data traffic.

The management of encryption/authentication protocols and SPIs is defined in [RFC7402]. The UDP encapsulation format and processing of HIP ESP traffic is described in Section 6.1 of [RFC7402].

5.12. RELAYED_ADDRESS and MAPPED_ADDRESS Parameters

While the type values are new, the format of the RELAYED_ADDRESS and MAPPED_ADDRESS parameters (Figure 12) is identical to REG_FROM, RELAY_FROM and RELAY_TO parameters. This document specifies only the use of UDP relaying, and, thus, only protocol 17 is allowed. However, future documents may specify support for other protocols.

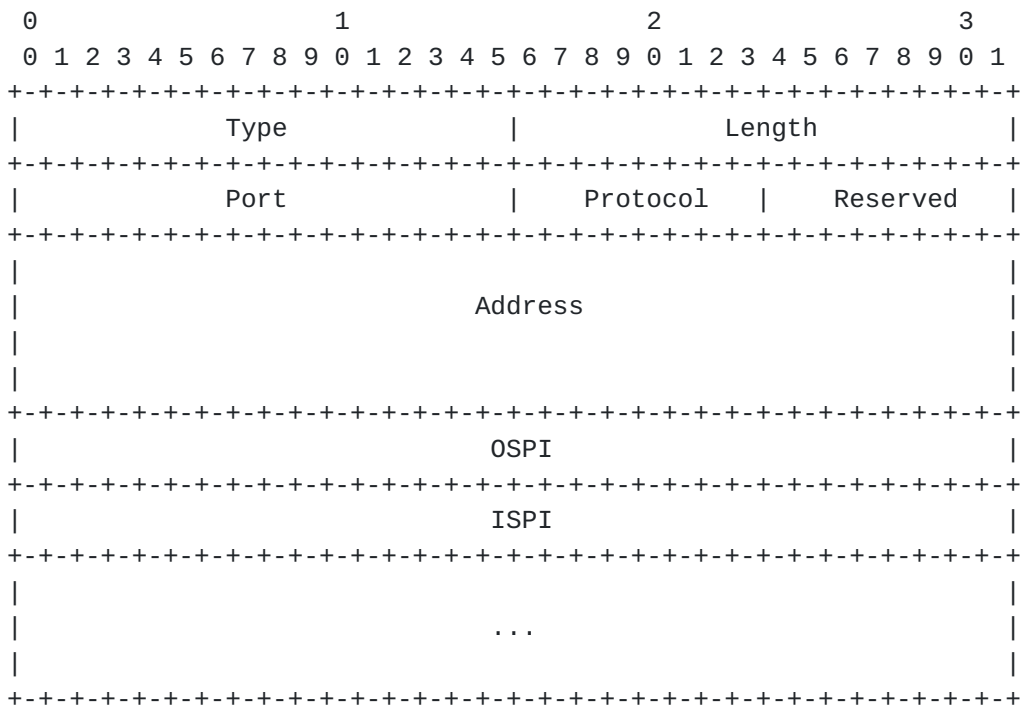


Type [TBD by IANA;
RELAYED_ADDRESS: 4650
MAPPED_ADDRESS: 4660]
Length 20
Port the UDP port number
Protocol IANA assigned, Internet Protocol number (17 for UDP)
Reserved reserved for future use; zero when sent, ignored when received
Address an IPv6 address or an IPv4 address in "IPv4-Mapped IPv6 address" format

Figure 12: Format of the RELAYED_ADDRESS and MAPPED_ADDRESS Parameters

5.13. PEER_PERMISSION Parameter

The format of the new PEER_PERMISSION parameter is shown in Figure 13. The parameter is used for setting up and refreshing forwarding rules and the permissions for data packets at the data relay. The parameter contains one or more sets of Port, Protocol, Address, Outbound SPI (OSPI), and Inbound SPI (ISPI) values. One set defines a rule for one peer address.

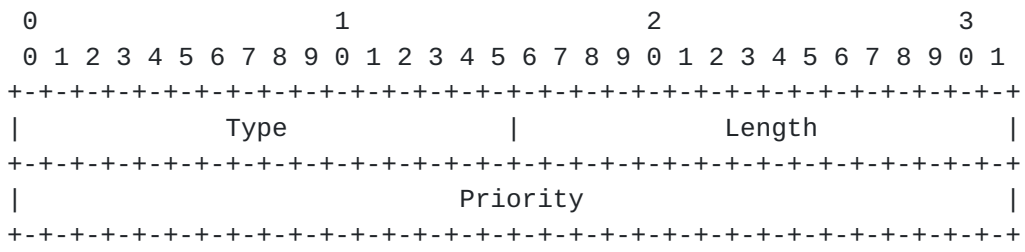


Type	[TBD by IANA; 4680]
Length	length in octets, excluding Type and Length
Port	the transport layer (UDP) port number of the peer
Protocol	IANA assigned, Internet Protocol number (17 for UDP)
Reserved	reserved for future use; zero when sent, ignored when received
Address	an IPv6 address, or an IPv4 address in "IPv4-Mapped IPv6 address" format, of the peer
OSPI	the outbound SPI value the registered host is using for the peer with the Address and Port
ISPI	the inbound SPI value the registered host is using for the peer with the Address and Port

Figure 13: Format of the PEER_PERMISSION Parameter

5.14. HIP Connectivity Check Packets

The connectivity request messages are HIP UPDATE packets containing a new CANDIDATE_PRIORITY parameter (Figure 14). Response UPDATE packets contain a MAPPED_ADDRESS parameter (Figure 12).

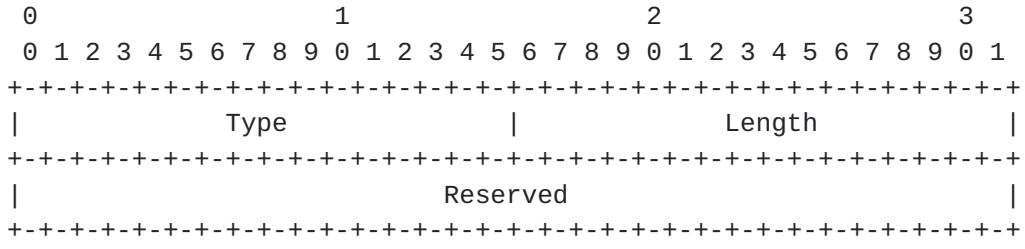


Type [TBD by IANA; 4700]
Length 4
Priority the priority of a (potential) peer reflexive candidate

Figure 14: Format of the CANDIDATE_PRIORITY Parameter

5.15. NOMINATE parameter

Figure 15 shows the NOMINATE parameter that is used to conclude the candidate nomination process.



Type [TBD by IANA; 4710]
Length 4
Reserved Reserved for future extension purposes

Figure 15: Format of the NOMINATE Parameter

6. Security Considerations

The security considerations are the same as in [RFC5770], but are repeated here for the sake of completeness.

6.1. Privacy Considerations

The locators are in plain text format in favor of inspection at HIP-aware middleboxes in the future. The current document does not specify encrypted versions of LOCATOR_SETs, even though it could be beneficial for privacy reasons to avoid disclosing them to middleboxes.

It is also possible that end-users may not want to reveal all locators to each other. For example, tracking the physical location

of a multihoming end-host may become easier if it reveals all locators to its peer during a base exchange. Also, revealing host addresses exposes information about the local topology that may not be allowed in all corporate environments. For these two reasons, an end-host may exclude certain host addresses from its LOCATOR_SET parameter. However, such behavior creates non-optimal paths when the hosts are located behind the same NAT. Especially, this could be problematic with a legacy NAT that does not support routing from the private address realm back to itself through the outer address of the NAT. This scenario is referred to as the hairpin problem [RFC5128]. With such a legacy NAT, the only option left would be to use a relayed transport address from a TURN server.

The use of HIP and data relays can be also useful for privacy purposes. For example, a privacy concerned Responder may reveal only its HIP relay server and Relayed candidates to Initiators. This same mechanism also protects the Responder against Denial-of-Service (DoS) attacks by allowing the Responder to initiate new connections even if its relays would be unavailable due to a DoS attack.

6.2. Opportunistic Mode

A HIP relay server should have one address per relay client when a HIP relay is serving more than one relay client and supports opportunistic mode. Otherwise, it cannot be guaranteed that the HIP relay server can deliver the I1 packet to the intended recipient.

6.3. Base Exchange Replay Protection for HIP Relay Server

In certain scenarios, it is possible that an attacker, or two attackers, can replay an earlier base exchange through a HIP relay server by masquerading as the original Initiator and Responder. The attack does not require the attacker(s) to compromise the private key(s) of the attacked host(s). However, for this attack to succeed, the Responder has to be disconnected from the HIP relay server.

The relay can protect itself against replay attacks by becoming involved in the base exchange by introducing nonces that the end-hosts (Initiator and Responder) are required to sign. One way to do this is to add ECHO_REQUEST_M parameters to the R1 and I2 packets as described in [HIP-MIDDLE] and drop the I2 or R2 packets if the corresponding ECHO_RESPONSE_M parameters are not present.

6.4. Demultiplexing Different HIP Associations

Section 5.1 of [RFC3948] describes a security issue for the UDP encapsulation in the standard IP tunnel mode when two hosts behind different NATs have the same private IP address and initiate

communication to the same Responder in the public Internet. The Responder cannot distinguish between two hosts, because security associations are based on the same inner IP addresses.

This issue does not exist with the UDP encapsulation of HIP ESP transport format because the Responder uses HITs to distinguish between different Initiators.

6.5. Reuse of Ports at the Data Relay

If the data relay uses the same relayed address and port (as conveyed in the RELAYED_ADDRESS parameter) for multiple registered hosts, it appears to all the peers, and their firewalls, that all the registered hosts using the relay are at the same address. Thus, a stateful firewall may allow packets pass from hosts that would not normally be able to send packets to a peer behind the firewall. Therefore, a HIP data relay SHOULD NOT re-use the port numbers. If port numbers need to be re-used, the relay SHOULD have a sufficiently large pool of port numbers and select ports from the pool randomly to decrease the chances of a registered host obtaining the same address that a another host behind the same firewall is using.

6.6. Amplification attacks

A malicious host may send an invalid list of candidates for its peer that are used for targeting a victim host by flooding it with connectivity checks. To mitigate the attack, this protocol adopts the ICE mechanism to cap the total amount of connectivity checks as defined in section Section 4.7.

6.7. Attacks against Connectivity Checks and Candidate Gathering

Section 13.1 in [I-D.ietf-ice-rfc5245bis] discusses about attacks against ICE connectivity checks. HIP bases its control plane security on Diffie-Hellman key exchange, public keys and Hashed Message Authentication codes, meaning that the mentioned security concerns do not apply to HIP either. The mentioned section discusses also of man-in-the-middle replay attacks that are difficult to prevent. The connectivity checks in this protocol are immune against replay attacks because a connectivity request includes a random nonce that the recipient must sign and send back as a response.

Section 13.2 in [I-D.ietf-ice-rfc5245bis] discusses attacks on server reflexive address gathering. Similarly here, if the DNS, a HIP relay or a HIP data relay server has been compromised, not much can be done. However, the case where attacker can inject fake messages (located on a shared network segment like Wifi) does not apply here.

HIP messages are integrity and replay protected, so it is not possible inject fake server reflexive address candidates.

Section 13.3 in [I-D.ietf-ice-rfc5245bis] discusses attacks on relayed candidate gathering. Similarly to ICE TURN servers, data relays require an authenticated base exchange that protects relayed address gathering against fake requests and responses. Further, replay attacks are not possible because the HIP base exchange (and also UPDATE procedure) is protected against replay attacks.

7. IANA Considerations

This section is to be interpreted according to [RFC5226].

This document updates the IANA Registry for HIP Parameter Types [RFC7401] by assigning new HIP Parameter Type values for the new HIP Parameters: RELAYED_ADDRESS, MAPPED_ADDRESS (defined in Section 5.12), and PEER_PERMISSION (defined in Section 5.13).

This document updates the IANA Registry for HIP NAT traversal modes [RFC5770] by assigning value for the NAT traversal mode ICE-HIP-UDP (defined in Section 5.4) This specification introduces a new keepalive Notify message type field NAT_KEEPALIVE.

This document defines additional registration types for the HIP Registration Extension [RFC8003] that allow registering with a HIP relay server for ESP relaying service: RELAY_UDP_ESP (defined in Section 4.1; and performing server reflexive candidate discovery: CANDIDATE_DISCOVERY (defined in Section 4.2).

ICE specifications discuss "Unilateral Self-Address Fixing" in section 17 in [I-D.ietf-ice-rfc5245bis]. This protocol is based on ICE, and thus the same considerations apply also here with one exception: this protocol does not hide binary IP addresses from application-level gateways.

8. Contributors

Marcelo Bagnulo, Philip Matthews and Hannes Tschofenig have contributed to [RFC5770]. This document leans heavily on the work in the RFC.

9. Acknowledgments

Thanks to Jonathan Rosenberg and the rest of the MMUSIC WG folks for the excellent work on ICE. In addition, the authors would like to thank Andrei Gurtov, Simon Schuetz, Martin Stiemerling, Lars Eggert, Vivien Schmitt, and Abhinav Pathak for their contributions and Tobias

Heer, Teemu Koponen, Juhana Mattila, Jeffrey M. Ahrenholz, Kristian Slavov, Janne Lindqvist, Pekka Nikander, Lauri Silvennoinen, Jukka Ylitalo, Juha Heinanen, Joakim Koskela, Samu Varjonen, Dan Wing, Tom Henderson, Alex Elsayed and Jani Hautakorpi for their comments to [RFC5770], which is the basis for this document.

This work has been partially funded by CyberTrust programme by Digile/Tekes in Finland.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC7401] Moskowitz, R., Ed., Heer, T., Jokela, P., and T. Henderson, "Host Identity Protocol Version 2 (HIPv2)", RFC 7401, DOI 10.17487/RFC7401, April 2015, <http://www.rfc-editor.org/info/rfc7401>.

[RFC8003] Laganier, J. and L. Eggert, "Host Identity Protocol (HIP) Registration Extension", RFC 8003, DOI 10.17487/RFC8003, October 2016, <http://www.rfc-editor.org/info/rfc8003>.

[RFC8004] Laganier, J. and L. Eggert, "Host Identity Protocol (HIP) Rendezvous Extension", RFC 8004, DOI 10.17487/RFC8004, October 2016, <http://www.rfc-editor.org/info/rfc8004>.

[RFC8046] Henderson, T., Ed., Vogt, C., and J. Arkko, "Host Mobility with the Host Identity Protocol", RFC 8046, DOI 10.17487/RFC8046, February 2017, <http://www.rfc-editor.org/info/rfc8046>.

[RFC5770] Komu, M., Henderson, T., Tschofenig, H., Melen, J., and A. Keranen, Ed., "Basic Host Identity Protocol (HIP) Extensions for Traversal of Network Address Translators", RFC 5770, DOI 10.17487/RFC5770, April 2010, <http://www.rfc-editor.org/info/rfc5770>.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, DOI 10.17487/RFC5389, October 2008, <http://www.rfc-editor.org/info/rfc5389>.

[RFC7402] Jokela, P., Moskowitz, R., and J. Melen, "Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP)", RFC 7402, DOI 10.17487/RFC7402, April 2015, <http://www.rfc-editor.org/info/rfc7402>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <http://www.rfc-editor.org/info/rfc4291>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[I-D.ietf-ice-rfc5245bis] Keranen, A., Holmberg, C., and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal", draft-ietf-ice-rfc5245bis-08 (work in progress), December 2016.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, DOI 10.17487/RFC2475, December 1998, <http://www.rfc-editor.org/info/rfc2475>.

10.2. Informative References

[RFC4423] Moskowitz, R. and P. Nikander, "Host Identity Protocol (HIP) Architecture", RFC 4423, DOI 10.17487/RFC4423, May 2006, <http://www.rfc-editor.org/info/rfc4423>.

[RFC5201] Moskowitz, R., Nikander, P., Jokela, P., Ed., and T. Henderson, "Host Identity Protocol", RFC 5201, DOI 10.17487/RFC5201, April 2008, <http://www.rfc-editor.org/info/rfc5201>.

[RFC5207] Stiemerling, M., Quittek, J., and L. Eggert, "NAT and Firewall Traversal Issues of Host Identity Protocol (HIP) Communication", RFC 5207, DOI 10.17487/RFC5207, April 2008, <http://www.rfc-editor.org/info/rfc5207>.

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, DOI 10.17487/RFC5766, April 2010, <http://www.rfc-editor.org/info/rfc5766>.

[RFC6538] Henderson, T. and A. Gurtov, "The Host Identity Protocol (HIP) Experiment Report", RFC 6538, DOI 10.17487/RFC6538, March 2012, <http://www.rfc-editor.org/info/rfc6538>.

[MMUSIC-ICE] Rosenberg, J., "Guidelines for Usage of Interactive Connectivity Establishment (ICE) by non Session Initiation Protocol (SIP) Protocols", Work in Progress, July 2008.

[RFC5128] Srisuresh, P., Ford, B., and D. Kegel, "State of Peer-to-Peer (P2P) Communication across Network Address Translators (NATs)", RFC 5128, DOI 10.17487/RFC5128, March 2008, <http://www.rfc-editor.org/info/rfc5128>.

[HIP-MIDDLE] Heer, T., Wehrle, K., and M. Komu, "End-Host Authentication for HIP Middleboxes", Work in Progress, February 2009.

[RFC3948] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, DOI 10.17487/RFC3948, January 2005, <http://www.rfc-editor.org/info/rfc3948>.

Appendix A. Selecting a Value for Check Pacing

Selecting a suitable value for the connectivity check transaction pacing is essential for the performance of connectivity check-based NAT traversal. The value should not be so small that the checks cause network congestion or overwhelm the NATs. On the other hand, a pacing value that is too high makes the checks last for a long time, thus increasing the connection setup delay.

The Ta value may be configured by the user in environments where the network characteristics are known beforehand. However, if the characteristics are not known, it is recommended that the value is adjusted dynamically. In this case, it is recommended that the hosts estimate the round-trip time (RTT) between them and set the minimum Ta value so that only two connectivity check messages are sent on every RTT.

One way to estimate the RTT is to use the time that it takes for the HIP relay server registration exchange to complete; this would give an estimate on the registering host's access link's RTT. Also, the I1/R1 exchange could be used for estimating the RTT, but since the R1 can be cached in the network, or the relaying service can increase the delay notably, this is not recommended.

Appendix B. Differences with respect to ICE

The protocol specified in this document follows the semantics of ICE as close as possible, and most of the differences are syntactical due to the use of a different protocol. In this section, we describe the differences to the ICE protocol.

- ICE operates at the application layer, whereas this protocol operates between transport and network layers, thus hiding the protocol details from the application.

- The STUN protocol is not employed. Instead, this protocol reuses the HIP control plane format in order simplify demultiplexing of different protocols. For example, the STUN binding response is replaced with a HIP UPDATE message containing an ECHO_REQUEST_SIGN parameter and the STUN binding response with a HIP UPDATE message containing an ECHO_RESPONSE_SIGNED parameter as defined in section Section 4.6.

- The TURN protocol is not utilized. Instead, this protocol reuses HIP relay servers for the same purpose.

- ICMP errors may be used in ICE to signal failure. In this protocol, HIP NOTIFY messages are used instead.

- Instead of the ICE username fragment and password mechanism for credentials, this protocol uses the HIT, derived from a public key, for the same purpose. The username fragments are "transient host identifiers, bound to a particular session established as part of the candidate exchange" [I-D.ietf-ice-rfc5245bis]. In HIP, a local public key and the derived HIT are considered long-term identifiers, and invariant across different host associations and different transport-layer flows.

- In ICE, the conflict when two communicating end-points take the same controlling role is solved using random values (so called tie-breaker value). In this protocol, the conflict is solved by the standard HIP base exchange procedure, where the host with the "larger" HIT switches to Responder role, thus changing also to controlled role.

- The ICE-CONTROLLED and ICE-CONTROLLING attributes are not included in the connectivity checks.

- The foundation concept is unnecessary in this protocol because only a single UDP flow for the IPsec tunnel will be negotiated.

- Frozen candidates are omitted for the same reason as foundation concept is excluded.

- Components are omitted for the same reason as foundation concept is excluded.

- This protocol supports only "full ICE" where the two communicating hosts participate actively to the connectivity checks, and the "lite" mode is not supported. This design decision follows the guidelines of ICE which recommends full ICE implementations. However, it should be noted that a publicly reachable Responder may refuse to negotiate the ICE mode as described in Section 4.7.2. This would result in a [RFC7401] based HIP base exchange tunneled over UDP followed ESP traffic over the same tunnel, without the connectivity check procedures defined in this document (in some sense, this mode corresponds to the case where two ICE lite implementations connect since no connectivity checks are sent).

- As the "ICE lite" is not adopted here and both sides are capable of ICE-HIP-HIP mode (negotiated during the base exchange), default candidates are not employed here.

- If the agent is using Diffserv Codepoint markings [RFC2475] in its media packets, it SHOULD apply those same markings to its connectivity checks.

- Unlike in ICE, the addresses are not XOR-ed in this protocol in order to avoid middlebox tampering.

- This protocol does not employ the ICE related address and related port attributes (that are used for diagnostic or SIP purposes).

Appendix C. Differences to Base Exchange and UPDATE procedures

This section gives some design guidance for implementers how the extensions in this protocol extends and differs from [RFC7401] and [RFC8046].

- Both control and data plane are operated on top of UDP, not directly on IP.

- A minimal implementation would conform only to Section 4.7.1 or Section 4.7.2, thus merely tunneling HIP control and data traffic over UDP. The drawback here is that it works only in the limited cases where the Responder has a public address.

- It is worth noting that while a rendezvous server [RFC8004] has not been designed to be used in NATted scenarios because it just relays the first I1 packet and does not employ UDP encapsulation, the HIP relay forwards all control traffic and, hence, is more suitable in NATted environments. Further, the data relay concept guarantees forwarding of data plane traffic also in the cases when the NAT penetration procedures fail.

- Registration procedures with a relay server are similar as with rendezvous server. However, a relay has different registration parameters than rendezvous because it offers a different service. Also, the relay includes also a REG_FROM parameter that informs the client about its server reflexive address. In the case of a data relay, it includes also a RELAYED_ADDRESS containing the relayed address for the client.

- In [RFC7401], the Initiator and Responder can start to exchange application payload immediately after the base exchange. While exchanging data immediately after a base exchange via a data relay would be possible also here, we follow the ICE methodology to establish a direct path between two hosts using connectivity checks. This means that there will be some additional delay after the base exchange before application payload can be transmitted. The same applies for the UPDATE procedure as the connectivity checks introduce some additional delay.

- In HIP without NAT traversal support, the base exchange acts as an implicit connectivity check, and the mobility and multihoming extensions support explicit connectivity checks. After a base exchange or UPDATE based connectivity checks, a host can use the associated address pair for transmitting application payload. In this extension, we follow the ICE methodology, where one end-point acting in the controlled role chooses the used address pair also on behalf of the other end-point acting in controlled role, which is different from HIP without NAT traversal support. Another difference is that the process of choosing an address pair is explicitly signaled using the nomination packets. The nomination process in this protocol supports only single address pair, and multihoming extensions are left for further study.

- The UPDATE procedure resembles the mobility extensions defined in [RFC8046]. The first UPDATE message from the mobile host is exactly the same as in the mobility extensions. The second UPDATE message from the peer host and third from the mobile host are different in the sense that they merely acknowledge and conclude the reception of the candidates through the relay. In other words, they do not yet test for connectivity (besides reachability through the HIP relay) unlike in the mobility extensions. The

idea is that connectivity check procedure follows the ICE specification, which is somewhat different from the HIP mobility extensions.

- The connectivity checks as defined in the mobility extensions [RFC8046] are triggered only by the peer of the mobile host. Since successful NAT penetration requires that both end-points test connectivity, both the mobile host and its peer host have to test for connectivity. In addition, this protocol validates also the UDP ports; the ports in the connectivity check must match with the response, as required by ICE.

- In HIP mobility extensions [RFC8046], an outbound locator has some associated state: UNVERIFIED mean that the locator has not been tested for reachability, ACTIVE means that the address has been verified for reachability and is being used actively, and DEPRECATED means that the locator lifetime has expired. In the subset of ICE specifications used by this protocol, an individual address candidate has only two properties: type and priority. Instead, the actual state in ICE is associated with candidate pairs rather than individual addresses. The subset of ICE specifications utilized by this protocol require the following attributes for a candidate pair: valid bit, nominated bit, base and the state of connectivity check. The connectivity checks have the following states: Waiting, In-progress, Succeeded and Failed. Handling of this state attribute requires some additional logic when compared to the mobility extensions since the state is associated with a local-remote address pair rather just a remote address, and, thus, the mobility and ICE states do not have an unambiguous one-to-one mapping.

- Credit-based authorization as defined in [RFC8046] could be used before candidate nomination has been concluded upon discovering working candidate pairs. However, this may result in the use of asymmetric paths for a short time period in the beginning of communications (similarly as in aggressive ICE nomination). Thus, support of credit-based authorization is left for further study.

Authors' Addresses

Ari Keranen
Ericsson
Hirsalantie 11
02420 Jorvas
Finland

Email: ari.keranen@ericsson.com

Jan Melen
Ericsson
Hirsalantie 11
02420 Jorvas
Finland

Email: jan.melen@ericsson.com

Miika Komu (editor)
Ericsson
Hirsalantie 11
02420 Jorvas
Finland

Email: miika.komu@ericsson.com