Host Identity Protocol
Internet-Draft
Intended status: Experimental
Expires: January 10, 2013

R. Hummen, Ed.
M. Henze
RWTH Aachen University,
Communication and Distributed
Systems Group
July 09, 2012

HIP Middlebox Puzzle Offloading and End-host Notification
draft-hummen-hip-middle-puzzle-00

Abstract

The Host Identity Protocol [RFC5201] is a secure signaling protocol with a cryptographic namespace. It provides the communicating peers with a cryptographic puzzle mechanism to protect against Denial of Service (DoS) attacks targeting its computation and memory overhead. This document specifies an extension that enables middleboxes to assist in the choice of the puzzle difficulty as well as in solving the puzzle on behalf of the host.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Notation

{x}	indicates that x is signed.
Initiator	is the host that initiates a HIP association (cf. HIP base protocol).
Responder	is the host that responds to the INITIATOR (cf. HIP base protocol).
-->	signifies "Initiator to Responder" communication.
<--	signifies "Responder to Initiator" communication.

Status of this Memo

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

This Internet-Draft will expire on January 10, 2013.

Table of Contents

1. Introduction

The Host Identity Protocol (HIP) [RFC5201] is a secure signaling protocol that introduces a cryptographic namespace for the identification and authentication of the communicating peers. As the protocol employs computationally expensive public-key-based operations in the protocol exchange, HIP provides the communicating peers with mechanisms to protect against Denial of Service (DoS) attacks targeting its computation and memory overhead. Specifically, the Responder in the protocol handshake may require the Initiator to solve a dynamically adjustable cryptographic puzzle. The puzzle enables the Responder to require a commitment of resources from the Initiator before performing computationally expensive operations and setting up association state.

While such a protection mechanism is generally desirable for protocols involving computationally expensive operations, the correct choice of the puzzle difficulty is hard. This is especially true in communication scenarios where the resources of the communicating peers are highly diverse. An example for such a scenario is a resource-constrained sensor node communicating with a comparably powerful modern desktop computer.

This document specifies an extension of the HIP signaling channel that enables middleboxes to participate in the handshake between two hosts in order to assist either in the choice of the puzzle difficulty or in solving the puzzle. To this end, the extension introduces additional signaling between hosts and middleboxes that are located on the communication path. The corresponding message exchanges and the newly introduced parameters are described in this document. Note, however, that the recommendation of specific puzzle difficulties is out-of-scope for this document as these are highly scenario and situation dependent.

2. Considerations concerning the HIP puzzle

The cryptographic puzzle mechanism used in HIP allows the responder to protect against maliciously induced computations as well as memory exhaustion. In a scenario without load, the Responder should set the puzzle difficulty K to 0. As a result, any value is a correct solution for the puzzle. Hence, the Initiator does not need to commit resources into solving the puzzle to continue the HIP handshake. However, when the Responder is under high load (e.g., due to an on-going attack), it may increase the puzzle difficulty K. The exact value of K should depend on the available resources of the Initiator in order to prevent the use of undemanding or excessively hard puzzles. However, IP addresses, HITs, and cipher suites that

are known to the Responder when setting the puzzle difficulty do not suffice to make this information available to the Responder.

Difficulty K	MSP430 16 MHz	Intel Core 2 Duo 2.4 GHz
5	0.15 sec	0.001 sec
10	4.92 sec	0.02 sec
15	127.06 sec	0.45 sec
20	4253.08 sec	14.90 sec

Table 1: Average time for solving puzzles with difficulty K for sensor-class (MSP430) and desktop-class (Core 2 Duo) devices over 50 measurements

Table 1 shows that high values of K (e.g., K = 20) require an excessive number of computations for sensor-class devices, while low puzzle difficulties (e.g., K = 5) only provide negligible protection against modern desktop-class attackers. This observation results in the following two obstacles when using a puzzle to protect the Responder against DoS attacks.

2.1. Resource-constrained Initiator

Assume a scenario where a resource-constrained device initiates a new handshake with an (Internet-connected) desktop-class Responder that is currently under high load (e.g., due to an attack). In the optimal case (i.e., without NATs), the Responder learns the IP address of the Initiator, its HIT, and the DH_GROUP_LIST. However, the provided information does not allow the Responder to deduce whether it is communicating with a resource-constrained or a comparably powerful device. Hence, the Responder has to guess the class of the Initiator and set the puzzle difficulty accordingly. If the Responder sets the puzzle difficulty for a desktop-class device, the puzzle is most likely unsolvable for sensor-class devices. Likewise, if the Responder assumes a sensor-class device, the puzzle does not protect against DoS attacks from a desktop-class device. As the latter choice negates the use of the puzzle mechanism, the Responder should always set the difficulty such that it protects against attacks from computationally powerful peer hosts.

In order to enable communication with a resource-constrained device despite the use of high puzzle difficulties, this document proposes a mechanism that allows resource-constrained Initiators to offload solving of the puzzle to on-path middleboxes.

2.2. Resource-constrained Responder

Assume a scenario where resource-constrained devices primarily communicate with each other. However, at the same time, they are directly accessible from the Internet via gateway nodes. An example for such a network topology may be a 6LowPAN-enabled sensor network that is equipped with a 6LBR (see Section 1.2 in [I-D.ietf-6lowpan-nd]).

If a HIP-enabled device initiates a new connection with a resource-constrained device that is currently under high load (e.g., due to an attack), the resource-constrained Responder cannot identify the capabilities of the peer similarly to the case described above. Hence, the Responder does not protect itself against malicious Internet hosts, if he sets a puzzle difficulty that is suitable for sensor-class devices. Contrarily, if he sets the puzzle difficulty too high, this prevents connections from benign sensor-class devices.

To allow communication with other legitimate resource-constrained devices, the resource-constrained Responder should use puzzle difficulties targeting other resource-constrained devices. However, to still protect against malicious desktop-class devices, this document introduces a mechanism that enables the on-path gateway to signal the Responder that traffic is routed via an untrusted network. This enables the Responder to set higher puzzle difficulties in case of communication where the capabilities of the peer host are uncertain.

3. Protocol Overview

This section gives an overview of the interaction between hosts and middleboxes that assist resource-constrained hosts in using the HIP puzzle mechanism. Specifically, this document describes how resource-constrained Initiators can offload solving of a puzzle to an on-path middlebox and how a middleboxes can signal resource-constrained Responders to set the puzzle difficulty for Internet hosts.

3.1. Assisting Resource-constrained Initiators

A resource-constrained Initiator may receive a puzzle that would require excessive computations. If energy, time, or availability constraints do not allow the Initiator to solve the puzzle itself, it can offload its computation to on-path middleboxes.

As puzzle offloading requires changes to the SOLUTION parameter, the offloading Initiator, the computing middlebox, and the verifying

Responder are required to support the extension described in this documents. Hence, end host (EOS) and middlebox puzzle offloading support (MOS) are negotiated in the R1 packet (see Figure 1). As a result of the negotiation, the Initiator learns if the puzzle offloading extension described in this document can be used when processing the received R1 packet.

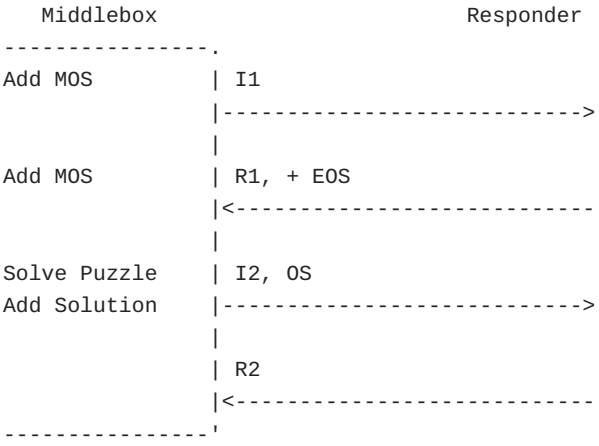


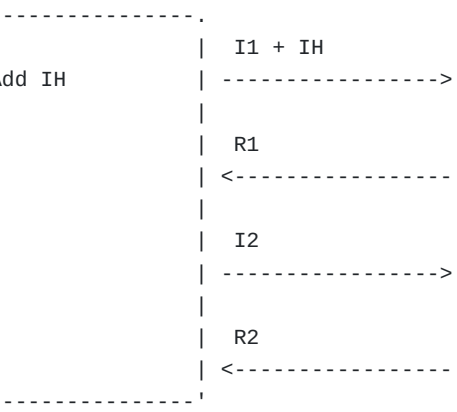
Figure 1

In the I2 packet, the OFFLOADED_SOLUTION replaces the SOLUTION parameter (see Figure 1). The OFFLOADED_SOLUTION has got the same parameter layout as the original SOLUTION parameter. However, it is placed in the unsigned part of the I2 packet. The Initiator echoes the puzzle parameters in the OFFLOADED_SOLUTION, while leaving the solution value blank. When receiving an OFFLOADED_SOLUTION parameter, an on-path middlebox computes the solution based on the parameter fields in the OFFLOADED_SOLUTION parameter and places the solution value in the blank solution field of the OFFLOADED_SOLUTION. The algorithm used to compute the puzzle solution can be derived from the HIT of the Responder contained in the HIP header. Note that the puzzle offloading extension is designed not to require additional state at either the initiator, the middlebox, or the responder.

3.2. Assisting Resource-constrained Responders

A resource-constrained device that is currently under high load may receive an initial handshake packet (I1). In order to protect against attacks from within the local (resource-constrained) network environment, the Responder SHOULD set a low puzzle difficulty by

default. To still protect against malicious Internet hosts, an on-path middlebox notifies the Responder about handshakes that originate from the Internet. Such a notification SHOULD trigger the Responder to set a higher puzzle difficulty for this specific handshake targeting the uncertain capabilities of the Internet-connected peer.



IH: Internet Host

Figure 2

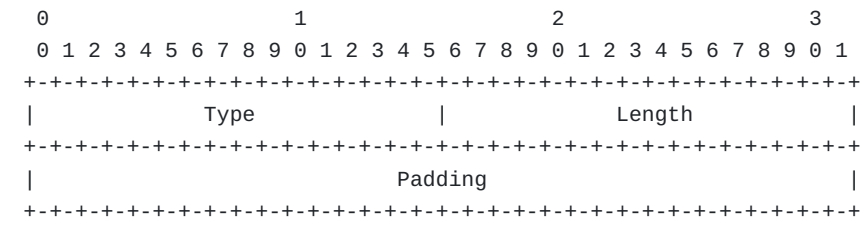
As shown in Figure 2, the middlebox notifies the Responder in the I1 packet that the current handshake originates from an Internet host. The Responder SHOULD then set the puzzle difficulty as to protect against an attack from a desktop-class device.

4. HIP Parameters

This HIP extension specifies four new HIP parameters that allow on-path middleboxes to assist in choosing a puzzle difficulty as well as in computing a puzzle solution on behalf of a host.

4.1. HOST_OFFLOADING_SUPPORT

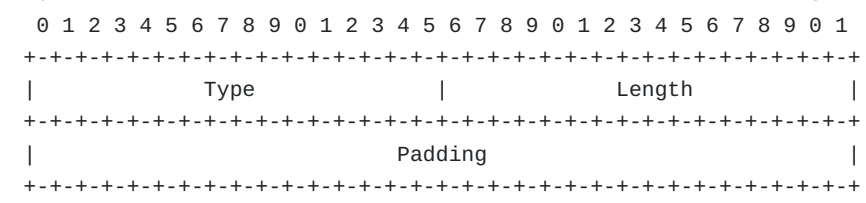
The Responder MAY append the HOST_OFFLOADING_SUPPORT parameter to an R1 packet in order to indicate the support of the puzzle offloading mechanism described in this document. The parameter is located in the signed part of the HIP control packet and is, therefore, bound to the host identity of the Responder.



Type 5600
Length 0

4.2. MIDDLEBOX_OFFLOADING_SUPPORT

A middlebox MAY append the MIDDLEBOX_OFFLOADING_SUPPORT parameter to an R1 packet in order to indicate the support of the puzzle offloading mechanism described in this document. The parameter is located in the unsigned part of the HIP control packet. Middleboxes SHOULD check for the existence of an MIDDLEBOX_OFFLOADING_SUPPORT in the R1 packet before adding the parameter in order to prevent multiple parameters of this type to be included in the R1 packet. Note, however, that this check SHOULD be done for reasons of space-efficiency and does not have an impact on the offloading mechanism itself. Middleboxes that support the puzzle offloading extension SHOULD NOT keep per-association state about their notifications.



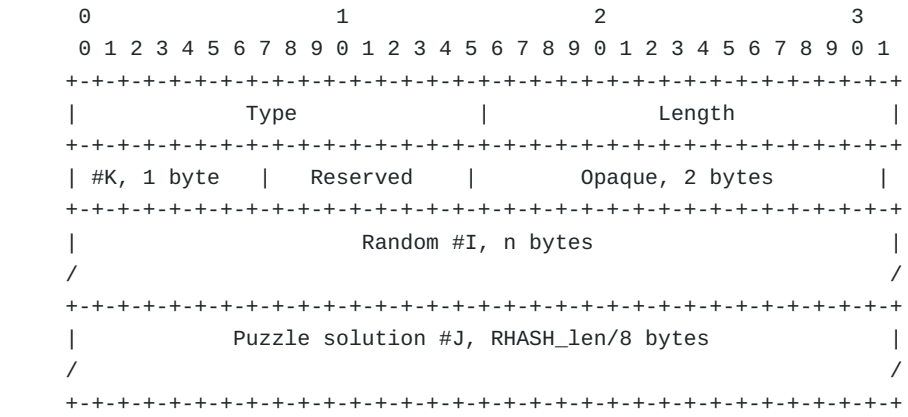
Type 64600
Length 0

4.3. OFFLOADED_SOLUTION

When receiving a HOST_OFFLOADING_SUPPORT and a MIDDLEBOX_OFFLOADING_SUPPORT parameter in the R1 packet, the Initiator MAY add the OFFLOADED_SOLUTION instead of the SOLUTION parameter in the I2 packet. In this case, the Initiator SHOULD skip the computation of the puzzle solution.

The structure and semantics of the OFFLOADED_SOLUTION parameter equal the SOLUTION parameter in [RFC5201]. However, the Initiator sets the #J to 0. This indicates that an on-path middlebox MUST compute the puzzle solution.

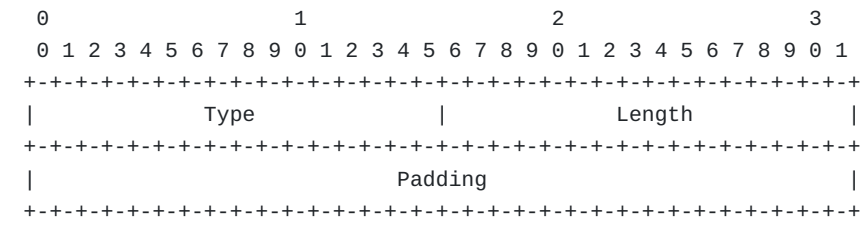
When receiving an I2 packet containing the OFFLOADED_SOLUTION parameter, an on-path middlebox that supports the puzzle offloading extension first inspects the #J. If #J is 0, the middlebox uses the echoed PUZZLE values in the OFFLOADED_SOLUTION parameter as well as the RHASH function to compute the puzzle solution. It then adds the computed solution to the #J of the OFFLOADED_SOLUTION parameter. Hence, the first middlebox supporting the puzzle offloading mechanism computes the puzzle solution on behalf of the Initiator. Note that the OFFLOADED_SOLUTION parameter is located in the unsigned part of the HIP packet. Hence, the modification of the parameter by the middlebox does not invalidate existing integrity protection mechanisms.



Type	64602
Length	4 + RHASH_len /4
#K	#K is the number of verified bits
Reserved	zero when sent, ignored when received
Opaque	copied unmodified from the received PUZZLE
Random #I	random number of size RHASH_len bits
Puzzle solution #J	random number of size RHASH_len bits

4.4. VIA_UNTRUSTED_NETWORK

A middlebox MAY append the VIA_UNTRUSTED_NETWORK parameter to an R1 packet in order to indicate that the handshake is routed via an untrusted network. As a result, the Responder SHOULD set the puzzle difficulty in the PUZZLE parameter to target the uncertain capabilities of the peer host.



Type 64604
Length 0

5. Security Considerations

This document describes a modification of the HIP puzzle mechanism used in the initial HIP handshake. The puzzle offloading extension replaces the signed SOLUTION parameter by the unsigned OFFLOADED_SOLUTION parameter. This allows an on-path attacker to increase the puzzle difficulty K. As a result, the middlebox has to commit additional resources for computing the puzzle solution for a higher difficulty than originally required by the Responder.

The MIDDLEBOX_OFFLOADING_SUPPORT parameter is not cryptographically bound to a specific middlebox that claims to support the extension and to take over the workload of computing the puzzle solution. Hence, an on-path attacker could use the new parameter to trick the Initiator into using the offloading extension described in this document, although it is not supported on the communication path or despite the fact that he is unwilling to compute the puzzle solution. As a result, the Responder would receive a blank, invalid puzzle solution and drop the I2 packet. However, the attacker could achieve the same result by simply dropping any of the handshake packets.

The VIA_UNTRUSTED_NETWORK parameter is not cryptographically bound to the middlebox that claims a specific handshake to originate from an untrusted network. Hence, an on-path attacker could trick the Responder into increasing the puzzle difficulty, although the peer host is within the local (resource-constrained) network environment. As a result, the Initiator would drop the resulting puzzle due to its excessive difficulty. However, the attacker could simply drop the I1 or the R1 packet in order to achieve the same result.

6. IANA Considerations

This document specifies four new HIP parameter types. The preliminary parameter type numbers are 5600, 64600, 64602, and 64604.

7. Acknowledgments

Thanks to Jens Hiller for commenting and helping to improve the quality of this document.

8. Changelog

8.1. Version 0

- Initial Version

9. Normative References

[I-D.ietf-6lowpan-nd] Shelby, Z., Chakrabarti, S., and E. Nordmark, "Neighbor Discovery Optimization for Low Power and Lossy Networks (6LoWPAN)", draft-ietf-6lowpan-nd-18 (work in progress), October 2011.

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

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

Authors' Addresses

Rene Hummen (editor)
RWTH Aachen University, Communication and Distributed Systems Group
Ahornstrasse 55
Aachen 52074
Germany

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

Martin Henze
RWTH Aachen University, Communication and Distributed Systems Group
Ahornstrasse 55
Aachen 52074
Germany

Email: henze@comsys.rwth-aachen.de
URI: http://www.comsys.rwth-aachen.de/team/martin-henze/