Internet Engineering Task Force	D.K. Kuptsov
Internet-Draft	A.G. Gurtov
Intended status: Informational	Helsinki Institute for Information Technology, Aalto University
Expires: September 15, 2011	D.Z. Zhang
	Huawei Technologies Co.,Ltd
	March 14, 2011

Hierarchical Host Identity Tags (HHIT) Verification Architecture
draft-kuptsov-hhit-05

Abstract

This document describes the architecture for hierarchical host identity tags (HHIT) vrification for HIP protocol.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

This document describes the architecture for hierarchical host identity tags (HHIT) vrification for Host Identity Protocol (HIP) RFC 5201 *[RFC5201]*.
The purpose of HHIT architecture is to enable online verification of flat identifiers (in a scalable way), such as Host Identity Tags (HIT), by corresponding organizations that are responsible for clients holding such identifiers. While one way of verifying whether HIT belongs to a client that is affiliated with some organization (or unit within organization) is to use certificates; such approach can be undesired because it (i) introduces high cost for certificate verification, and (ii) does not directly allow certificate status verification (consider the situation when private key of a particular host was stolen and firewall enforcing certificate verification does not check the revocation status of host's certificate).

2. Structure of HHIT

The following are the goals of HHIT: (i) allow any on the path security gateway to distinguish to which authority the identifier belongs, and later ask corresponding authority whether given HHIT is valid; (ii) prevent misuse of HHIT by attackers (specifically, the design allows to prevent replaying and constructing "fake" HHITs that will enable attackers to bypass the security gateways).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            OID                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            HHIT                               |
+                                                               |
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+


+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type            |C|             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        ENC_HHIT_TIMESTAMP                     |
+                                                               +
|                                                               |
+                                                               +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Padding (4 bytes)                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The structure of hierarchical HHIT:

* OID is organization identifier that depending of the application of HHIT can be globally unique (e.g., assigned by Internet Assigned Numbers Authority (IANA)), or unique within certain scope (e.g., within organization and assigned on per department or unit granularity). Length of OID is 32 bits.

* HHIT is an output of a pseudo-random function (PRF) under one-time secret key and input taken as a concatenation of OID and flat identifier (HIT): HHIT=PRF(OID || HIT, secret) The length of HHIT field is 96 bits to guarantee sufficient level of security.

* ENC_HHIT_TIMESTAMP parameter guarantees freshness of HHIT, it contains the timestamp when particular HHIT was generated. This field is encrypted (using symmetric encryption function) under the same one-time secret as HHIT: ENC_HHIT_TIMESTAMP = ENC(HHIT || #epoch || padding (96 bits), secret), where HHIT is as described above, #epoch is a timestamp indicating the time when HHIT was constructed, and secret is the next yet unused secret key from a key pull, assigned to a client by its authority. Because the usage

of block cipher is assumed for encryption, the length of ENC_HHIT_TIMESTAMP field is a multiple of the block size of a particular encryption algorithm. Length of #epoch is 64 bits to allow encode timestamp in microseconds. As a result, the length of ENC_HHIT_TIMESTAMP when used together with AES-CBC algorithm, is 2*128 bits.

Because total length of OID||HHIT||ENC_HHIT_TIMESTAMP exceeds reserved 128 bits for source address in HIP protocol, the Sender's Host Identity Tag should contain only OID||HHIT, while ENC_HHIT_TIMESTAMP should be carried as mandatory HIP parameter in I1 packet.

3. Use case

```
                       Register HHIT (offline)
+----------------------+------------------>+-------------+
|                      |                   |  Domain 1   |
|Client (from domain 1)|    Secret keys    |  authority  |
+----------------------+<------------------+-------+-----+
                       |                  HHIT /\  | OK
                       |                       |   v
                       |   I1               +---+---------+
                       +------------------>|  Security   |-->...
                       +------------------>|  gateway    |
                       |   I1               +---+---+-----+
                       |                  HHIT |   /\
                       |   Register HHIT       v   |  Ok
+----------------------+------------------>+-------------+
|Client (from domain 2)|                   |  Domain 2   |
|                      |    Secret keys    |  authority  |
+----------------------+<------------------+-------+-----+
```

Next we describe a possible use case - access control with HHIT:

*Each end-host that belongs to some organization, or organizational unit, constructs its HIT (using the procedure described in RFC 5201 *[RFC5201]*), and registers it in an offline manner in its organizational repository. Depending on the application, the registration itself can involve authentication, e.g. using passwords, certificates, biometric information, passport, etc. Upon verification, domain authority generates set of one-time-passwords (the number of such passwords again depends on the application needs), and for each secret s populates its database with the following records: HHIT = PRF(OID || HIT, s) Domain authority then returns list of secrets to client over secure channel (how this is achieved is out of scope).

* When a client wants to access the service that is behind security gateway, it chooses next unused one-time secret "unused secret" and constructs the HHIT as PRF(OID || HIT, "unused secret"), it also takes the current #epoch "now" and constructs ENC_HHIT_TIMESTAMP parameter as ENC(HHIT || "now", "unused secret").

* Every I1 packet then contains: sender's Host Identity Tag field as (OID || HHIT), also parameter ENC_HHIT_TIMESTAMP is added such that domain authority can verify the freshness of HHIT.

* When security gateway receives such I1 packet, it will look-up the domain authority using OID found in the sender's Host Identity Tag, and submit OID, HHIT, and ENC_HHIT_TIMESTAMP to corresponding domain authority. Security gateway will buffer I1 until it will receive (positive or negative) response from corresponding domain authority.

* Last, when domain authority receives OID, HHIT, and ENC_HHIT_TIMESTAMP for verification it looks up for the proper secret using HHIT as index. If the record was not found, the domain authority immediately responds to a gateway that information is not valid. Else, domain authority attempts to decrypt ENC_HHIT_TIMESTAMP field to find #epoch. It also retrieves the last registered I1 timestamp (if any) -- "#epoch last". To mitigate possible replays, for every received I1 packet domain authority should check the timestamp found in ENC_HHIT_TIMESTAMP, and the timestamp of previously seen I1 packet for the same source. Optimally, timestamp found in received I1 packet should be grater then the last registered timestamp, i.e., the timestamps for the same source should be monotonically increasing #epoch > "#epoch last". However, consecutive I1s can be reordered, i.e., #epoch < "#epoch last". In this case if "#epoch last" - #epoch > Delta, the HHIT will be considered as invalid, and negative response will be sent to security gateway.

* Security gateway will make a forwarding decision regarding particular buffered I1 packet based on the response it receives from domain authority: if the response is negative I1 packet is dropped, otherwise the state will be created and I1 will be forwarded. Note, for consequent R1, I2, R2 packets the forwarding decisions by security gateway are done solely based on the stored internal state: if it exists the packets are forwarded, otherwise dropped.

4. Experimental observations

Experimental results (mean/median)				
Parameter	lambda=1	lambda=10	lambda=100	lambda=1000
Droprate(%)	0.016/0.014	0.017/0.015	0.68/0.76	0.909/0.901
Verification duration(ms)	1304/902.6	1339/896.7	1586/982.2	3826/2204
Queue size	2.4/2	18.04/17.0	301.5/283.0	1009/1022

To grasp what would be the performance implications, we measured HHIT verification using 2 DHTs deployed in the Internet and single security gateway. Each DHT was mimic single domain authority. We generated storms of I1 packets towards security gateway, using exponential distribution for interarrival times with different lambda parameter 1,10,100,1000 which corresponded to average interarrival times 1000, 100, 10, 1 (ms) respectively. We then measured 3 parameters: (a) drop rate (due to failed verification when I1 arrived out off order, or connection timeout), (b) HHIT verification duration (ms) and (c) queue buildup (number of I1 packets waiting for verification request completion). We present mean and median statistics for these parameters in the table below.

5. Security Considerations

We mentioned earlier that for every sent I1 packet, sender picks next unused one-time secret to produce HHIT and ENC_HHIT_TIMESTAMP. However, it can be sufficient for domain authority and particular client to share a single secret which is rotated every time T (where T can be on the scale of days).
The Delta threshold should be relatively small to prevent replays. Thus, Delta should be of order of few hundred milliseconds to guarantee sufficient level of security.

6. References

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

Authors' Addresses

Dmitriy Kuptsov Kuptsov Helsinki Institute for Information Technology, Aalto University PO Box 19215 Espoo, 00076 Aalto Finland
EMail: dmitriy.kuptsov@hiit.fi

Andrei Gurtov Gurtov Helsinki Institute for Information Technology, Aalto University PO Box 19215 Espoo, 00076 Aalto Finland EMail: gurtov@hiit.fi

Dacheng Zhang Zhang Huawei Technologies Co.,Ltd PO Box 19215 Beijing, China EMail: zhangdacheng@huawei.com