Network Working Group   M. Bagnulo
Internet-Draft   UC3M
Intended status: Standards Track   October 17, 2006
Expires: April 20, 2007

Hash Based Addresses (HBA)
draft-ietf-shim6-hba-02

Status of this Memo

Copyright Notice

Abstract

This memo describes a mechanism to provide a secure binding between the multiple addresses with different prefixes available to a host within a multihomed site. The main idea is that information about the multiple prefixes is included within the addresses themselves. This is achieved by generating the interface identifiers of the addresses of a host as hashes of the available prefixes and a random number. Then, the multiple addresses are generated by prepending the different prefixes to the generated interface identifiers. The

result is a set of addresses, called Hash Based Addresses (HBAs), that are inherently bound.

Table of Contents

1. Introduction

In order to preserve inter-domain routing system scalability, IPv6 sites obtain addresses from their Internet Service Providers. Such addressing strategy significantly reduces the amount of routes in the global routing tables, since each ISP only announces routes to its own address blocks, rather than announcing one route per customer site. However, this addressing scheme implies that multihomed sites will obtain multiple prefixes, one per ISP. Moreover, since each ISP only announces its own address block, a multihomed site will be reachable through a given ISP if the ISP prefix is contained in the destination address of the packets. This means that, if an established communication needs to be routed through different ISPs during its lifetime, addresses with different prefixes will have to be used. Changing the address used to carry packets of an established communication exposes the communication to numerous attacks, as described in [8], so security mechanisms are required to provide the required protection to the involved parties. This memo describes a tool that can be used to provide protection against some of the potential attacks, in particular against future/ premeditated attacks (a.k.a. time shifting attacks in [9]).

It should be noted that, as opposed to the mobility case where the addresses that will be used by the mobile node are not known a priori, the multiple addresses available to a host within the multihomed site are pre-defined and known in advance in most of the cases. The mechanism proposed in this memo takes advantage of this address set stability, and provides a secure binding between all the addresses of a node in a multihomed site. The mechanism does so without requiring the usage of public key cryptography, providing a cost efficient alternative to public key cryptography based schemes.

This memo describes a mechanism to provide a secure binding between the multiple addresses with different prefixes available to a host within a multihomed site. The main idea is that information about the multiple prefixes is included within the addresses themselves. This is achieved by generating the interface identifiers of the addresses of a host as hashes of the available prefixes and a random number. Then, the multiple addresses are obtained by prepending the different prefixes to the generated interface identifiers. The result is a set of addresses, called Hash Based Addresses (HBAs), that are inherently bound. A cost efficient mechanism is available to determine if two addresses belong to the same set, since given the prefix set and the additional parameters used to generate the HBA, a single hash operation is enough to verify if an HBA belongs to a given HBA set. No public key operations are involved in the verification process. In addition, it should also be noted that it is not required that all interface identifiers of the addresses of an

HBA set are equal, preserving some degree of privacy through changes in the addresses used during the communications.

2. Overview

2.1. Threat Model

The threat analysis for the multihoming problem is described in [8].This analysis basically identifies attacks based on redirection of packets by a malicious attacker towards addresses that do not belong to the multihomed node. There are essentially two type of redirection attacks: communication hijacking and flooding attacks. communication hijacking attacks are about an attacker stealing on-going and/or future communications from a victim. Flooding attacks are about redirecting the traffic generated by a legitimate source towards a third party, flooding it. The HBA solution provides full protection against the communication hijacking attacks and limited protection against flooding attacks. Residual threats are described in the security cosniderations section.

2.2. Overview

The basic goal of the HBA mechanism is to securely bind together multiple IPv6 addresses that belong to the same multihomed host. This allows rerouting of traffic without worrying that the communication is being redirected to an attacker. The technique that is used is to inlcude a hash of the permitted prefixes in the low order bits of the IPv6 address.

So, eliding some details, say the available prefixes are A, B, C, and D, the host would generate a prefix list P consisting of (A,B,C,D) and a random number M. Then it would generate the new addresses:

A || H(M || A || P)

B || H(M || B || P)

C || H(M || C || P)

D || H(M || D || P)

Thus, given one valid address out of the group and the prefix list P and the random number M it is possible to determine whether another address is part of the group by computing the hash and checking against the low order bits.

2.3. Motivations for the HBA design

The design of the HBA technique was dirven by the following considerations:

First of all, the goal of HBA is to provide a secure binding between the IPv6 address used as identifier by the upper layer protocols and the alternative locators available in the multihomed node, so that redirection attacks are prevented.

Second, in order to achieve such protection, the selected approach was to include security information in the identifer itself, instead of relying in third trusted parties to secure the binding, such as the ones based on repositories or Public Key Infrastructure. this decision was driven by deployment considerations i.e. the cost of deploying the third trusted party infrastucture.

Third, application support considerations described in [14] resulted in selecting routable IPv6 addresses to be used as identifiers. Hence, security information is stuffed within the interface identifier part of the IPv6 address.

Fourth, performance considerations as described in [15] motivated the usage of a hash based approach as oposed to a public key based approach based on pure CGA, in order to avoid imposing the performance of public key operations for every communication in multihomed environments. The HBA appraoch presented in this document present a cheaper alternative that is attractive to many common usage cases. Note that the HBA appraoch and the CGA appraoches are not mutually exclusive and that it is possible to generate addresses that are both CGA and HBA providing the benefits of both approaches if needed.

3. CGA compatibility considerations

As described in previous section, the HBA technique uses the interface identifier part of the IPv6 address to encode information about the multiple prefixes available to a multihomed host. However, the interface identifier is also used to carry cryptographic information when Cryptographic Generated Addresses [1] are used. Therefore, conflicting usages of the interface identifier bits may result if this is not taken into account during the HBA design. There are at least two valid reasons to provide CGA-HBA compatibility:

First, the current Secure Neighbor Discovery specification [2] uses the CGAs defined in [1] to prove address ownership. If HBAs are not

compatible with CGAs, then nodes using HBAs for multihoming wouldn't be able to do Secure Neighbor Discovery using the same addresses (at least the parts of SeND that require CGAs). This would imply that nodes would have to choose between security (from SeND) and fault tolerance (from shim6). In addition to SeND, there are other protocols that are considering to benefit from the advantages offered by the CGA scheme, such as mobility support protocols [10]. Those protocols would also become incompatible with HBAs if HBAs are not compatible with CGAs.

Second, CGAs provide additional features that cannot be achieved using only HBAs. In particular, because of its own nature, the HBA technique only supports a predetermined prefix set that is known at the time of the generation of the HBA set. No additions of new prefixes to this original set are supported after the HBA set generation. In most of the cases relevant for site multihoming, this is not a problem because the prefix set available to a multihomed set is not very dynamic. New prefixes may be added in a multihomed site when a new ISP is available, but the timing of those events are rarely in the same time scale than the lifetime of established communications. It is then enough for many situations that the new prefix is not available for established communications and that only new communications benefit from it. However, in the case that such functionality is required, it is possible to use CGAs to provide it. This approach clearly requires that HBA and CGA approaches are compatible. If this is the case, it then would be possible to create HBA/CGA addresses that support CGA and HBA functionality simultaneously. The inputs to the HBA/CGA generation process will be both a prefix set and a public key. In this way, a node that has established a communication using one address of the CGA/HBA set can tell its peer to use the HBA verification when one of the addresses of its HBA/CGA set is used as locator in the communication or to use CGA (public/private key based) verification when a new address that does not belong to the HBA/CGA set is used as locator in the communication.

So, because of the aforementioned reasons, it is a goal of the HBA design to define HBAs in a way that they are compatible with CGAs as defined in [1] and their usages described in [2] (Consequently, to understand the rest of this note, the reader should be familiar with the CGA specification defined in [1]). This means that it must be possible to generate addresses that are both an HBA and a CGA i.e. that the interface identifier contains cryptographic information of CGA and the prefix-set information of an HBA. The CGA specification already considers the possibility of including additional information into the CGA generation process through the usage of Extension Fields in the CGA Parameter Data Structure. It is then possible to define a Multi-Prefix Extension for CGA so that the prefix set information is

included in the interface identifier generation process.

Even though a CGA compatible approach is adopted, it should be noted that HBAs and CGAs are different concepts. In particular, the CGA is inherently bound to a public key, while a HBA is inherently bound to a prefix set. This means that a public key is not strictly required to generate an HBA. Because of that, we define three different types of addresses:

- CGA-only addresses: These are addresses generated as specified in [1] without including the Multi-Prefix Extension. They are bound to a public key and to a single prefix (contained in the basic CGA Parameter Data Structure). These addresses can be used for SeND [2] and if used for multihoming, their application will have to be based on the public key usage.

- CGA/HBA addresses: These addresses are CGAs that include the Multi-Prefix Extension in the CGA Parameters Data Structure used for their generation. These addresses are bound to a public key and a prefix set and they provide both CGA and HBA functionalities. They can be used for SeND as defined in [2] and for any usage defined for HBA (such as a shim6 protocol)

- HBA-only addresses: These addresses are bound to a prefix set but they are not bound to a public key. Because CGA compatibility, the CGA Parameter Data Structure will be used for their generation, but a random nonce will be included in the Public Key field instead of a public key. These addresses can be used for HBA based multihoming protocols, but they cannot be used for SeND.

4. Multi-Prefix Extension for CGA

The Multi-Prefix Extension has the following TLV format as defined in [6] :

```
     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
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |          Extension Type       |   Extension Data Length       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |P|                          Reserved                           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    +                           Prefix[1]                           +
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    +                           Prefix[2]                           +
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    .                               .                               .
    .                               .                               .
    .                               .                               .
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    +                           Prefix[n]                           +
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Ext Type: 16-bit type identifier of the Multi-Prefix Extension (TBD IANA) (Meanwhile, the 0x12 value is recommended for trails)

Ext Len: 16-bit unsigned integer. Length of the Extension in octets, not including the first 4 octets.

P flag: Set if a public key is included in the Public Key field of the CGA Parameter Data Structure. Reset if a additional Modifier bits are included in the CGA Parameter Data Structure.

Reserved: 31-bit reserved field. MUST be initialized to zero, and ignored upon receipt.

Prefix[1...n]: Vector of 64-bit prefixes, numbered 1 to n.

5. HBA-Set Generation

The HBA generation process is based on the CGA generation process defined in section 4 of [1]. The goal is to require the minimum amount of changes to the CGA generation process.

The CGA generation process has three inputs: a 64-bit subnet prefix,

a public key (encoded in DER as an ASN.1 structure of the type SubjectPublicKeyInfo), and the security parameter Sec.

The main difference between the CGA generation and the HBA generation is that while a CGA can be generated independently, all the HBAs of a given HBA set have to be generated using the same parameters, which implies that the generation of the addresses of an HBA set will occur in a coordinated fashion. In this memo, we will describe a mechanism to generate all the addresses of a given HBA set. The generation process of each one of the HBA address of an HBA set will be heavily based in the CGA generation process defined in [1]. More precisely, the HBA set generation process will be defined as a sequence of lightly modified CGA generations.

The changes required in the CGA generation process when generating a single HBA are the following: First, the Multi-Prefix Extension has to be included in the CGA Parameters Data Structure. Second, in the case that the address being generated is an HBA-only address, a random nonce (encoded in DER as an ASN.1 structure of the type SubjectPublicKeyInfo) will have to be used as input instead of a valid public key.

The resulting HBA-set generation process is the following:

The inputs to the HBA generation process are:
- A vector of n 64-bit prefixes
- A Sec parameter, and
- In the case of the generation of a set of HBA/CGA addresses a public key is also provided as input (not required when generating HBA-only addresses)

The output of the HBA generation process are:
- An HBA-set
- their respective CGA Parameters Data Structures

The steps of the HBA-set generation process are:

1. Multi-Prefix Extension generation. Generate the Multi-Prefix Extension with the format defined in section 3. Include the vector of n 64-bit prefixes in the Prefix[1...n] fields. The Ext Len field value is (n*8 + 4). If a public key is provided, then the P flag is set. Otherwise, the P flag is reset.

2. Modifier generation. Generate a Modifier as a random or pseudorandom 128-bit value. If a public key has not been provided as an input, generate the Extended Modifier as a 384-bit random or pseudorandom value. Encode the Extended Modifier value as a RSA key in a DER-encoded ASN.1 structure of the type

SubjectPublicKeyInfo defined in the Internet X.509 certificate profile [3].

3. Concatenate from left to right the Modifier, 9 zero octets, the encoded public key or the encoded Extended Modifier (if no public key was provided) and the Multi-Prefix Extension. Execute the SHA-1 algorithm on the concatenation. Take the 112 leftmost bits of the SHA-1 hash value. The result is Hash2.

4. Compare the 16*Sec leftmost bits of Hash2 with zero. If they are all zero (or if Sec=0), continue with step (5). Otherwise, increment the modifier by one and go back to step (3).

5. Set the 8-bit collision count to zero.

6. For i=1 to n do

   6.1. Concatenate from left to right the final Modifier value, Prefix[i], the collision count, the encoded public key or the encoded Extended Modifier (if no public key was provided) and the Multi-Prefix Extension. Execute the SHA-1 algorithm on the concatenation. Take the 64 leftmost bits of the SHA-1 hash value. The result is Hash1[i].

   6.2. Form an interface identifier from Hash1[i] by writing the value of Sec into the three leftmost bits and by setting bits 6 and 7 (i.e., the "u" and "g" bits) both to zero.

   6.3. Generate address HBA[i] by concatenating Prefix[i] and the 64-bit interface identifier to form a 128-bit IPv6 address with the subnet prefix to the left and interface identifier to the right as in a standard IPv6 address [4].

   6.4. Perform duplicate address detection if required. If an address collision is detected, increment the collision count by one and go back to step (6). However, after three collisions, stop and report the error.

   6.5. Form the CGA Parameters Data Structure that corresponds to HBA[i] by concatenating from left to right the final modifier value, Prefix[i], the final collision count value, the encoded public key or the encoded Extended Modifier and the Multi-Prefix Extension.

[Note: most of the steps of the process are taken from [1]]

6. HBA verification

6.1. Verification that a particular HBA address corresponds to a given CGA Parameter Data Strcuture

HBAs are constructed as a CGA Extension, so a properly formated HBA and its correspondent CGA Parameter Data Structure will successfully finish the verification process described in section 5 of [1]. Such verification is useful when the goal is the verification of the binding between the public key and the HBA.

6.2. Verification that a particular HBA address belongs tto the HBA set associated to a given CGA Parameter Data Strcuture

For multihoming applications, it is also relevant to verify if a given HBA address belongs to a certain HBA set. An HBA set is identified by a CGA Parameter Data structure that contains a Multi-Prefix Extension. So, it is then needed to verify if an HBA belongs to the HBA set defined by a CGA Parameter Data Structure. It should be noted that it may be needed to verify if an HBA belongs to the HBA set defined by the CGA Parameter Data Structure of another HBA of the set. If this is the case, the CGA verification process as defined in [1] will fail, because the prefix included in the Subnet Prefix field of the CGA Parameter Data Structure will not match with the one of the HBA that is being verified. However, this not means that this HBA does not belong to the HBA set. In order to address this issue, it is only required to verify that the HBA prefix is included in prefix set defined in the Multi-Prefix Extension, and if this is the case, then substitute the prefix included in the Subnet Prefix field by the prefix of the HBA, and then perform the CGA verification process defined in [1].

So, the process to verify that an HBA belongs to an HBA set determined by a CGA Parameter Data Structure is called HBA verification and it is the following:

The inputs to the HBA verification process are:
- o An HBA
- o An CGA Parameter Data Structure

The steps of the HBA verification process are the following:

1. Verify that the 64-bit HBA prefix is included in the prefix set of the Multi-Prefix Extension. If it is not included, the verification fails. If it is included, replace the prefix contained in the Subnet Prefix field of the CGA Parameter Data Structure by the 64-bit HBA prefix.

2. Run the verification process described in section 5 of [1] with the HBA and the new CGA Parameters Data Structure (including the Multi-Prefix Extension) as inputs. The steps of the process are included below, extracted from [1]

   2.1. Check that the collision count in the CGA Parameters Data Structure is 0, 1 or 2. The CGA verification fails if the collision count is out of the valid range.

   2.2. Check that the subnet prefix in the CGA Parameters Data Structure is equal to the subnet prefix (i.e., the leftmost 64 bits) of the address. The CGA verification fails if the prefix values differ. [Note: This step is trivially successful because step 1]

   2.3. Execute the SHA-1 algorithm on the CGA Parameters Data Structure. Take the 64 leftmost bits of the SHA-1 hash value. The result is Hash1.

   2.4. Compare Hash1 with the interface identifier (i.e., the rightmost 64 bits) of the address. Differences in the three leftmost bits and in bits 6 and 7 (i.e., the "u" and "g" bits) are ignored. If the 64-bit values differ (other than in the five ignored bits), the CGA verification fails.

   2.5. Read the security parameter Sec from the three leftmost bits of the 64-bit interface identifier of the address. (Sec is an unsigned 3-bit integer.)

   2.6. Concatenate from left to right the modifier, 9 zero octets, and the public key, and any extension fields [in this case, the Multi-Prefix Extension will be included, at least] that follow the public key in the CGA Parameters data structure. Execute the SHA-1 algorithm on the concatenation. Take the 112 leftmost bits of the SHA-1 hash value. The result is Hash2.

   2.7. Compare the 16*Sec leftmost bits of Hash2 with zero. If any one of them is non-zero, the CGA verification fails. Otherwise, the verification succeeds. (If Sec=0, the CGA verification never fails at this step.)

7. Example of HBA application to a multihoming scenario

In this section, we will describe a possible application of the HBA technique to IPv6 multi-homing.

We will consider the following scenario: a multihomed site obtains Internet connectivity through two providers ISPA and ISPB. Each provider has delegated a prefix to the the multihomed site (PrefA::/nA and PrefB::/nb respectively). In order to benefit from multihoming, the hosts within the multihomed site will configure multiple IP addresses, one per available prefix. The resulting configuration is depicted in the next figure.

```
               +-------+
               | Host2 |
               |IPHost2|
               +-------+
                   |
                   |
               (Internet)
                /      \
               /        \
         +------+      +------+
         | ISPA |      | ISPB |
         |      |      |      |
         +------+      +------+
            |             |
             \           /
              \         /
         +---------------------+
         | multihomed site     |
         | PA::/nA             |
         | PB::/nB    +------+ |
         |            |Host1 | |
         |            +------+ |
         +---------------------+
```

We assume that both Host1 and Host2 support the shim6 protocol.

Host2 in not located in a multihomed site, so there is no need for him to create HBAs (it must be able to verify them though, in order to support the shim6 protocol, as we will describe next.)

Host1 is located in the multihomed site, so it will generate its addresses as HBAs. In order to do that, it needs to execute the HBA-set generation process as detailed in Section 4 of this memo. The inputs of the HBA-set generation process will be: a prefix vector containing the two prefixes available in its link i.e. PA:LA::/64

and PB:LB::/64, a Sec parameter value, and optionally a public key. In this case we will assume that a public key is provided so that we can also illustrate how a renumbering event can be supported when HBA/CGA addresses are used (see the sub-section referring to dynamic address set support). So, after executing the HBA-set generation process, Host1 will have: an HBA-set consisting in two addresses i.e. PA:LA:iidA and PB:LB:iidB with their respective CGA Parameter Data Structures i.e. CGA_PDS_A and CGA_PDS_B. Note that iidA and iidB are different but both contain information about the prefix set available in the multihomed site.

We will next consider a communication between Host1 and Host2. Assume that both ISPs of the multihomed site are working properly, so any of the available addresses in Host1 can be used for the communication. Suppose then that the communication is established using PA:LA:iidA and IPHost2 for Host1 and Host2 respectively. So far, no special shim6 support has been required, and PA:LA:iidA is used as any other global IP address

Suppose that at a certain moment one of the hosts involved in the communication decides that multihoming support is required in this communication (this basically means that one of the hosts involved in the communication desires enhanced fault tolerance capabilities for this communication, so that if an outage occurs, the communication can be rehomed to an alternative provider).

At this moment, the shim6 protocol Host-Pair Context establishment exchange will be perfomed between the two hosts (see [7].). In this exchange, Host1 will send CGA_PDS_A to Host2.

After the reception of CGA_PDS_A, Host2 will verify that the received CGA Parameter Data Structure corresponds to the address being used in the communication PA:LA:iidA. this means that Host2 will execute the HBA verification process described in Section 5 of this memo with PA:LA:iidA and CGA_PDS_A as inputs. In this case, the verification will succeed since the CGA Parameter Data Structure and the addresses used in the verification match.

As long as there are no outages affecting the communication path through ISPA, packets will continue flowing. If a failure affects the path through ISPA, Host1 will attempt to re-home the communication to an alternative address i.e. PB:LB:iidB. For that, after detecting the outage, Host1 will inform Host2 about the alternative address. Host2 will verify that the new address belongs to the HBA set of the initial address. For that, Host2 will execute the HBA verification process with the CGA Parameter Data Structure of the original address (i.e. CGA_PDS_A) and the new address (i.e. PB:LB:iidB) as inputs. The verification process will succeed because

PB:LB::/64 has been included in the Multi-Prefix Extension during the HBA-set generation process. Additional verifications may be required to prevent flooding attacks (see the comments about flooding attacks prevention in the Security Considerations section of this memo).

Once the new address is verified, it can be used as an alternative locator to re-home the communication, while preserving the original address (PA:LA:iidA) ad identifier for the upper layers. This means that following packets will be addressed to/from this new address. Note that no additional HBA verification is required for the following packets, since the new valid address can be stored in Host2.

Eventually, the communication will end and the associated shim6 context information will be discarded.

In this example, only the HBA capabilities of the Host1 addresses were used. In other words, neither the public key included in the CGA Parameter Data Structure nor its correspondent private key was used in the protocol. In the following section we will consider a case where its usage is required.

7.1. Dynamic Address Set Support

In the previous section we have presented the mechanisms that allow a host to use different addresses of a pre-determined set to exchange packets of a communication. The set of addresses involved was pre-determined and known when the communication was initiated. To achieve such functionality, only HBA functionalities of the addresses were needed. In this section we will explore the case where the goal is to exchange packets using additional addresses that were not known when the communication was established. An example of such situation is for instance when a new prefix is available in a site after a renumbering event. In this case, the hosts that have the new address available may want to use it in communications that were established before the renumbering event. In this case, HBA functionalities of the addresses are not enough and CGA capabilities are to be used.

Consider then the previous case of the communication between Host1 and Host2. Suppose that the communication is up and running, as described earlier. Host1 is using PA:LA:iidA and Host2 is using IPHost2 to exchange packets. Now suppose that a new address, PC:LC: addC is available in Host1. Note that this address is just a regular IPv6 address, and it is neither an HBA nor a CGA. Host1 wants to use this new address in the existent communication with Host2. It should be noted that the HBA mechanism described in the previous section cannot be used to verify this new address, since this address does not belong to the HBA set (since the prefix was not available at the

moment of the generation of the HBA set). This means that alternative verification mechanisms will be needed.

In order to verify this new address, CGA capabilities of PA:LA:iidA are used. Note that the same address is used, only that the verification mechanism is different. So, if Host1 wants to use PC:LC:addC to exchange packets in the established communication, it will use message of the shim6 protocol, conveying the new address, PC:LC:addC, and this message will be signed using the private key corresponding to the public key contained in CGA_PDS_A. When Host2 receives the message, it will verify the signature using the public key contained in the CGA Parameter Data Structure associated with the address used for establishing the communication i.e. CGA_PDS_A and PA:LA:iidA respectively. Once that the signature is verified, the new address (PC:LC:addC) can be used in the communication.

8. DNS considerations

HBA sets can be generated using any prefix set. Actually, the only particularity of the HBA is that they contain information about the prefix set in the interface identifier part of the address in the form of a hash, but no assumption about the properties of prefixes used for the HBA generation is made. This basically means that depending on the prefixes used for the HBA set generation, it may or may not be recommended to publish the resulting (HBA) addresses in the DNS.

9. IANA considerations

This document defines a new CGA Extension, the Multi-Prefix Extension. This extension has been assigned the CGA Extension Type value TBD (IANA).

10. Security considerations

The goal of HBAs is to create a group of addresses that are securely bound, so that they can be used interchangeably when communicating with a node. If there is no secure binding between the different addresses of a node, a number of attacks are enabled, as described in [8]. It particular, it would possible for an attacker to redirect the communications of a victim to an address selected by the attacker, hijacking the communication. When using HBAs, only the addresses belonging to an HBA set can be used interchangeably, limiting the addresses that can be used to redirect the communication to a well, pre-determined set, that belongs to the original node

involved in the communication. So, when using HBAs, a node that is communicating using address A can redirect the communication to a new address B if and only if B belongs to the same HBA set than A.

This means that if an attacker wants to redirect communications addressed to address HBA1 to an alternative address IPX, the attacker will need to create a CGA Parameters data structure that generates an HBA set that contains both HBA1 and IPX.

In order to generate the required HBA set, the attacker needs to find a CGA Parameter data structure that fulfills the following conditions:

- o the prefix of HBA1 and the prefix of IPX are included in the Multi-Prefix Extension
- o HBA1 is included in the HBA set generated.

(this assumes that it is acceptable for the attacker to redirect HBA1 to any address of the prefix of IPX).

The remaining fields that can be changed at will by the attacker in order to meet the above conditions are: the Modifier, other prefixes in the Multi-Prefix Extension and other extensions. In any case, in order to obtain the desired HBA set, the attacker will have to use a brute force attack, which implies the generation of multiple HBA sets with different parameters (for instance with a different Modifier) until the desired conditions are meet. The expected number of times that the generation process will have to be repeated until the desired HBA set is found is exponentially related with the number of bits containing hash information included in the interface identifier of the HBA. Since 59 of the 64 bits of the interface identifier contain hash bits, then the expected number of generations that will have to be performed by the attacker are O(2^59).

The protection against brute force attacks can be improved increasing the Sec parameter. A non zero Sec parameter implies that steps 3-4 of the generation process will be repeated O(2^(16*Sec)) times (expected number of times). If we assimilate the cost of repeating the steps 3-4 to the cost of generating the HBA address, we can estimate the number of times that the generation is to be repeated in O(2^(59+16*Sec)).

10.1. Security considerations when using HBAs in the shim6 protocol

In this section we will analyze the security provided by HBAs in the context of a shim6 protocol as described in section 6 of this memo.

First of all, it must be noted that HBAs cannot prevent man-in-the-middle (hereafter MITM) attacks. This means, that in the scenario

described in Section 6, if an attacker is located along the path between Host1 and Host2 during the lifetime of the communication, the attacker will be able to change the addresses used for the communication. This means that he will be able to change the addresses used in the communication, adding or removing prefixes at his will. However, the attacker must make sure that the CGA Parameter Data Structure and the HBA set is changed accordingly. This essentially means that the attacker will have to change the interface identifier part of the addresses involved, since a change in the prefix set will result in different interface identifiers of the addresses of the HBA set, unless the appropriate Modifier value is used (which would require O(2(59+16*Sec)) attempts). So, HBA don't provide MITM attacks protection, but a MITM attacker will have to change the address used in the communication in order to change the prefix set valid for the communication.

HBAs provide protection against time shifting attacks [8], [9]. In the multihoming context, an attacker would perform a time-shifted attack in the following way: an attacker placed along the path of the communication will modify the packets to include an additional address as a valid address for the communication. Then the attacker would leave the on-path location, but the effects of the attack would remain (i.e. the address would still be considered as a valid address for that communication). Next we will present how HBAs can be used to prevent such attacks.

If the attacker is not on-path when the initial CGA Parameter Data Structure is exchanged, his only possibility to launch a redirection attack is to fake the signature of the message for adding new addresses using CGA capabilities of the addresses. This implies discovering the public key used in the CGA Parameter Data Structure and then cracking the key pair, which doesn't seem feasible. So in order to launch a redirection attack, the attacker needs to be on-path when the CGA Parameter Data Structure is exchanged, so he can modify it. Now, in order to launch the redirection attack, the attacker needs to add his own prefix in the prefix set of the CGA Parameter Data Strcutre. We have seen in the previous section that there are two possible approaches for this:

1. Find the right Modifier value, so that the address initially used in the communication is contained in the new HBA set. The cost of this attack is O(2(59+16*Sec)) iterations of the generation process, so it is deemed unfeasible
2. Use any Modifier value, so that the address initially used in the communication is probably not included in the HBA set. In this case, the attacker must remain on-path, since he needs to rewrite the address carried in the packets (if not the endpoints will notice a change in the address used in the communication). This essentially means that the attacker cannot launch a time-shifted

attack, but he must be a full time man-in-the-middle.

So, the conclusion is that HBAs provide protection against time-shifted attacks

HBAs do not provide complete protection against flooding attacks, However, HBAs make very difficult to launch a flooding attack towards a specific address. It is possible though, to launch a flooding attack against a prefix.

Suppose that an attacker has easy access to a prefix PX::/nX and that he wants to launch a flooding attack to a host located in the address P:iid. The attack would consist in establishing a communication with a server S and requesting a heavy flow from it. Then simply redirect the flow to P:iid, flooding the target. In order to perform this attack the attacker need to generate an HBA set including P and PX in the prefix set and that the resulting HBA set contains P:iid. In order to do this, the attacker need to find the appropriate Modifier value. The expected number of attempts required to find such Modifier value is O(2(59+16*Sec)), as presented earlier. So, we can conclude that such attack is not feasible.

However, the target of a flooding attack is not limited to specific hosts, but it can also be launched against other element of the infrastructure, such as router or access links. In order to do that, the attacker can establish a communication with a server S and request a download of a heavy flow. Then, the attacker redirects the communication to any address of the target network. Even if the target address is not assigned to any host, the flow will flood the access link of the target site, and the site access router will also suffer the overload. Such attack cannot be prevented using HBAs, since the attacker can easily generate an HBA set using his own prefix and the target network prefix. In order to prevent such attacks, additional mechanisms are required, such as reachability tests.

10.2. Privacy Considerations

HBAs can be used as RFC 3041 [5]. addresses. If a node wants to use temporary addresses, it will need to periodically generate new HBA sets. The effort required for this operation depends on the Sec parameter value. If Sec=0, then the cost of generating a new HBA set is similar to the cost of generating a random number i.e. one iteration of the HBA set generation procedure. However, if Sec>0, then the cost of generating an HBA set is significantly increased, since it required O(2(16*Sec)) iterations of the generation process. In this case, depending on the frequency of address change required, the support for RFC 3041 address may be more expensive.

10.3. Interaction with IPSec.

In the case that both IPSec and CGA/HBA address are used simultaneously, it is possible that two public keys are available in a node, one for IPSec and another one for the CGA/HBA operation. In this case, an improved security can be achieved by verifying that the keys are related somehow, (in particular if the same key is used for both purposes).

10.4. SHA-1 Dependency Considerations.

Recent attacks to currently used hash functions have motivated a considerable amount of concern in the Internet community. The recommended approach [12] [13] to deal with this issue is first to analyze the impact of these attacks on the different Internet protocols that use hash functions and second to make sure that the different Internet protocols that use hash functions are capable of migrating to an alternative (more secure) hash function without a major disruption in the Internet operation.

The aforementioned analysis for CGAs and its extensions (including HBAs) is performed in [11]. The conclusion of the analysis is that the security of the protocols using CGAs and its extensions is not affected by the recently available attacks against hash functions. In spite of that an update to the CGA specification [1] to enable the support of alternative hash functions is proposed in [11]. In case that the proposed modifications are adopted for CGAs and that new mechanisms for generating CGAs are defined using alternative hash functions, HBA generation/verification mehtods will need to be produced compliant with the new CGA generation/verification methods defined.

11. Contributors

This document was originally produced of a MULTI6 design team consisting of (in alphabetical order): Jari Arkko, Marcelo Bagnulo, Iljitsch van Beijnum, Geoff Huston, Erik Nordmark, Margaret Wasserman, and Jukka Ylitalo.

12. Acknowledgments

The initial discussion about HBA benefited from contributions from Alberto Garcia-Martinez, Tuomas Aura and Arturo Azcorra.

The HBA-set generation and HBA verification processes described in this document contain several steps extracted from [1].

Jari Arkko, Matthew Ford, Francis Dupont, Mohan Parthasarathy, Pekka Savola, Brian Carpenter, Eric Rescorla and Sam Hartman have reviewed this document and provided valuable comments.

The text included in section 2.2 Overview was provided by Eric Rescorla.

We would also like to thanks Francis Dupont for providing the first implementation of HBA

13. Change Log

To be removed prior publication

13.1. Changes from draft-ietf-shim6-hba-01 to draft-ietf-multi6-hba-02

A new section SHA-1 Dependency Considerations has been added in the Security Considerations Section (addressing Eric Rescorla (SecDir) comment)

A new Overview section containing a Threat model subsection, a general description subsection and a motivations subsection has been added (addressing Eric Rescorla (SecDir) comment)

Modified section of HBA verification in order to improve readability

13.2. Changes from draft-ietf-shim6-hba-00 to draft-ietf-multi6-hba-01

Changed the format of the Multi-Prefix extension to make it compliant with the generic TLV format proposed for CGA extensions

Added IANA considerations section

Added DNS considerations section

13.3. Changes from draft-ietf-multi6-hba-00 to draft-ietf-shim6-hba-00

Editorial changes

13.4. Changes from draft-bagnulo-multi6dt-hba-00 to draft-ietf-multi6-hba-00

Added "Example of HBA application to a multihoming scenario" section

Added Privacy Considerations section

Added flooding attacks comments in the Security Considerations

section

Added the Multi-Prefix extension in step 6.1 of the HBA-set generation process

Added the Security considerations when using HBAs in a multi6 protocol sub-section in the Security Considerations section

Added Ext type value recommended for trials

Changed the name of the draft

Some rewording

14. References

14.1. Normative References

[1] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, April 2004.

[2] Arkko, J., Kempf, J., Sommerfeld, B., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, July 2004.

[3] Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.

[4] Hinden, R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, April 2003.

[5] Narten, T. and R. Draves, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 3041, January 2001.

[6] Bagnulo, M. and J. Arkko, "Cryptographically Generated Addresses (CGA) Extension Field Format", draft-bagnulo-shim6-cga-ext-02 (work in progress), October 2005.

[7] Nordmark, E., "Level 3 multihoming shim protocol", draft-ietf-shim6-proto-01 (work in progress), October 2005.

14.2. Informative References

[8] Nordmark, E., "Threats relating to IPv6 multihoming solutions", RFC 4218, October 2005.

[9] Nikander, P., Arkko, J., Aura, T., Montenegro, G., and E.

Nordmark, "Mobile IP version 6 Route Optimization Security Design Background", RFC 4225, December 2005.

[10] Haddad, W., Madour, L., Arkko, J., and F. Dupont, "Applying Cryptographically Generated Addresses to Optimize MIPv6 (CGA-OMIPv6)", draft-haddad-mip6-cga-omipv6-02 (work in progress), June 2004.

[11] Bagnulo, M. and J. Arkko, "Support for Multiple Hash Algorithms in Cryptographically Generated Addresses (CGAs)", draft-bagnulo-multiple-hash-cga-00 (work in progress), June 2006.

[12] Hoffman, P. and B. Schneier, "Attacks on Cryptographic Hashes in Internet Protocols", RFC 4270, November 2005.

[13] Bellovin, S. and E. Rescorla, "Deploying a new hash algorithm", 2005 September.

[14] Nordmark, E., "Multi6 Application Referral Issues", draft-nordmark-multi6dt-refer-00 (work in progress), October 2004.

[15] Bagnulo, M., Garcia-Martinez, A., and A. Azcorra, "Efficient Security for IPv6 Multihoming.", ACM Computer Communications Review Vol 35 n 2, April 2005.

Author's Address

Marcelo Bagnulo
Universidad Carlos III de Madrid
Av. Universidad 30
Leganes, Madrid 28911
SPAIN

Phone: 34 91 6249500
Email: marcelo@it.uc3m.es
URI: http://www.it.uc3m.es

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).