

IPv6 Maintenance (6man) Working Group
Internet-Draft
Obsoletes: [rfc4941](#) (if approved)
Intended status: Standards Track
Expires: September 6, 2018

F. Gont
SI6 Networks / UTN-FRH
S. Krishnan
Ericsson Research
March 5, 2018

Privacy Extensions for Stateless Address Autoconfiguration in IPv6
draft-gont-6man-rfc4941bis-00

Abstract

Nodes use IPv6 stateless address autoconfiguration to generate addresses using a combination of locally available information and information advertised by routers. Addresses are formed by combining network prefixes with an interface identifier. This document describes an extension that causes nodes to generate global scope addresses from interface identifiers that change over time. Changing the interface identifier (and the global scope addresses generated from it) over time makes it more difficult for eavesdroppers and other information collectors to identify when different addresses used in different transactions actually correspond to the same node.

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 6, 2018.

Copyright Notice

Copyright (c) 2018 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
1.1.	Terminology	3
1.2.	Problem Statement	3
2.	Background	5
2.1.	Extended Use of the Same Identifier	5
2.2.	Possible Approaches	6
3.	Protocol Description	8
3.1.	Assumptions	8
3.2.	Generation Of Randomized Interface Identifiers	9
3.2.1.	Randomized IIDs	9
3.2.2.	Hash-based generation of randomized interface identifiers	9
3.3.	Generating Temporary Addresses	11
3.4.	Expiration of Temporary Addresses	12
3.5.	Regeneration of Randomized Interface Identifiers	13
3.6.	Deployment Considerations	14
4.	Implications of Changing Interface Identifiers	16
5.	Defined Constants	17
6.	Future Work	18
7.	Security Considerations	19
8.	Significant Changes from RFC RFC4941	20
9.	Acknowledgements	21
10.	References	22
10.1.	Normative References	22
10.2.	Informative References	23
	Authors' Addresses	25

1. Introduction

Stateless address autoconfiguration [[RFC4862](#)] defines how an IPv6 node generates addresses without the need for a DHCPv6 server.

The security and privacy implications of such addresses have been discussed in great detail in [[RFC7721](#)], [[RFC7217](#)], and [RFC7707](#).

[Section 2](#) provides background information on the issue. [Section 3](#) describes a procedure for generating alternate interface identifiers and global scope addresses. [Section 4](#) discusses implications of changing interface identifiers. The term "global scope addresses" is used in this document to collectively refer to "Global unicast addresses" as defined in [[RFC4291](#)] and "Unique local addresses" as defined in [[RFC4193](#)].

1.1. 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](#)].

The terms "Public address", "stable address", "temporary address", "constant IID", "Stable IID", and "Temporary IID" are to be interpreted as specified in [[RFC7721](#)].

1.2. Problem Statement

Addresses generated using Stateless address autoconfiguration [[RFC4862](#)] contain an embedded interface identifier, which remains stable over time. Anytime a fixed identifier is used in multiple contexts, it becomes possible to correlate seemingly unrelated activity using this identifier.

The correlation can be performed by

- o An attacker who is in the path between the node in question and the peer(s) it is communicating to, and can view the IPv6 addresses present in the datagrams.
- o An attacker who can access the communication logs of the peers with which the node has communicated.

Since the identifier is embedded within the IPv6 address, which is a fundamental requirement of communication, it cannot be easily hidden. This document proposes a solution to this issue by generating interface identifiers which vary over time.

Note that an attacker, who is on path, may be able to perform significant correlation based on

- o The payload contents of the packets on the wire
- o The characteristics of the packets such as packet size and timing

Use of temporary addresses will not prevent such payload based correlation.

2. Background

This section discusses the problem in more detail, provides context for evaluating the significance of the concerns in specific environments and makes comparisons with existing practices.

2.1. Extended Use of the Same Identifier

The use of a non-changing interface identifier to form addresses is a specific instance of the more general case where a constant identifier is reused over an extended period of time and in multiple independent activities. Anytime the same identifier is used in multiple contexts, it becomes possible for that identifier to be used to correlate seemingly unrelated activity. For example, a network sniffer placed strategically on a link across which all traffic to/from a particular host crosses could keep track of which destinations a node communicated with and at what times. Such information can in some cases be used to infer things, such as what hours an employee was active, when someone is at home, etc. Although it might appear that changing an address regularly in such environments would be desirable to lessen privacy concerns, it should be noted that the network prefix portion of an address also serves as a constant identifier. All nodes at (say) a home, would have the same network prefix, which identifies the topological location of those nodes. This has implications for privacy, though not at the same granularity as the concern that this document addresses. Specifically, all nodes within a home could be grouped together for the purposes of collecting information. If the network contains a very small number of nodes, say just one, changing just the interface identifier will not enhance privacy at all, since the prefix serves as a constant identifier.

One of the requirements for correlating seemingly unrelated activities is the use (and reuse) of an identifier that is recognizable over time within different contexts. IP addresses provide one obvious example, but there are more. Many nodes also have DNS names associated with their addresses, in which case the DNS name serves as a similar identifier. Although the DNS name associated with an address is more work to obtain (it may require a DNS query) the information is often readily available. In such cases, changing the address on a machine over time would do little to address the concerns raised in this document, unless the DNS name is changed as well (see [Section 4](#)).

Web browsers and servers typically exchange "cookies" with each other [[COOKIES](#)]. Cookies allow web servers to correlate a current activity with a previous activity. One common usage is to send back targeted advertising to a user by using the cookie supplied by the browser to

identify what earlier queries had been made (e.g., for what type of information). Based on the earlier queries, advertisements can be targeted to match the (assumed) interests of the end-user.

The use of a constant identifier within an address is of special concern because addresses are a fundamental requirement of communication and cannot easily be hidden from eavesdroppers and other parties. Even when higher layers encrypt their payloads, addresses in packet headers appear in the clear. Consequently, if a mobile host (e.g., laptop) accessed the network from several different locations, an eavesdropper might be able to track the movement of that mobile host from place to place, even if the upper layer payloads were encrypted.

The security and privacy implications of IPv6 addresses are discussed in detail in [[RFC7721](#)], [[RFC7707](#)], and [[RFC7217](#)].

2.2. Possible Approaches

One way to avoid having a stable non-changing address is to use DHCPv6[DHCPv6] for obtaining addresses. Section 12 of [[DHCPv6](#)] discusses the use of DHCPv6 for the assignment and management of "temporary addresses", which are never renewed and provide the same property of temporary addresses described in this document with regards to the privacy concern.

Another approach, compatible with the stateless address autoconfiguration architecture, would be to change the interface identifier portion of an address over time. Changing the interface identifier can make it more difficult to look at the IP addresses in independent transactions and identify which ones actually correspond to the same node, both in the case where the routing prefix portion of an address changes and when it does not.

Many machines function as both clients and servers. In such cases, the machine would need a DNS name for its use as a server. Whether the address stays fixed or changes has little privacy implication since the DNS name remains constant and serves as a constant identifier. When acting as a client (e.g., initiating communication), however, such a machine may want to vary the addresses it uses. In such environments, one may need multiple addresses: a "stable" address and public address registered in the DNS, that is used to accept incoming connection requests from other machines, and a "temporary" address used to shield the identity of the client when it initiates communication. These two cases are roughly analogous to telephone numbers and caller ID, where a user may list their telephone number in the public phone book, but disable the display of its number via caller ID when initiating calls.

On the other hand, a machine that functions only as a client may want to employ only temporary addresses for public communication.

To make it difficult to make educated guesses as to whether two different interface identifiers belong to the same node, the algorithm for generating alternate identifiers must include input that has an unpredictable component from the perspective of the outside entities that are collecting information.

[I-D.gont-6man-non-stable-iids] specifies requirements for temporary addresses. This document specifies a number of algorithms for generating temporary addresses that comply with the aforementioned requirements.

3. Protocol Description

The goal of this section is to define procedures that:

1. Do not result in any changes to the basic behavior of addresses generated via stateless address autoconfiguration [[RFC4862](#)].
2. Create temporary addresses based on an unpredictable interface identifier for the purpose of initiating outgoing sessions. These temporary addresses would be used for a short period of time (hours to days) and would then be deprecated. Deprecated address can continue to be used for already established connections, but are not used to initiate new connections. New temporary addresses are generated periodically to replace temporary addresses that expire, with the exact time between address generation a matter of local policy.
3. Produce a sequence of temporary global scope addresses from a sequence of interface identifiers that appear to be random in the sense that it is difficult for an outside observer to predict a future address (or identifier) based on a current one and it is difficult to determine previous addresses (or identifiers) knowing only the present one.
4. By default, generate one address for each prefix to be employed for stateless address autoconfiguration.

3.1. Assumptions

The following algorithm assumes that for a given temporary address, an implementation can determine the prefix from which it was generated. When a temporary address is deprecated, a new temporary address is generated. The specific valid and preferred lifetimes for the new address are dependent on the corresponding lifetime values set for the prefix from which it was generated.

Finally, this document assumes that when a node initiates outgoing communication, temporary addresses can be given preference over stable addresses (if available), when the device is configured to do so. [[RFC6724](#)] mandates implementations to provide a mechanism, which allows an application to configure its preference for temporary addresses over stable addresses. It also allows for an implementation to prefer temporary addresses by default, so that the connections initiated by the node can use temporary addresses without requiring application-specific enablement. This document also assumes that an API will exist that allows individual applications to indicate whether they prefer to use temporary or stable addresses and override the system defaults.

3.2. Generation Of Randomized Interface Identifiers

The following subsections specify some possible algorithms for generating temporary interface identifiers that comply with the requirements in [[I-D.gont-6man-non-stable-iids](#)].

3.2.1. Randomized IIDs

One possible approach would be to select a pseudorandom number of the appropriate length. A node employing this algorithm should generate IIDs as follows:

1. Obtain a random number (see [[RFC4086](#)] for randomness requirements for security)
2. The Interface Identifier is obtained by taking as many bits from the aforementioned random number (obtained in the previous step) as necessary.

We note that [[RFC4291](#)] requires that the Interface IDs of all unicast addresses (except those that start with the binary value 000) be 64 bits long. However, the method discussed in this document could be employed for generating Interface IDs of any arbitrary length, albeit at the expense of reduced entropy (when employing Interface IDs smaller than 64 bits).

3. The resulting Interface Identifier SHOULD be compared against the reserved IPv6 Interface Identifiers [[RFC5453](#)] [[IANA-RESERVED-IID](#)] and against those Interface Identifiers already employed in an address of the same network interface and the same network prefix. In the event that an unacceptable identifier has been generated, a new interface identifier should be generated, by repeating the algorithm from the first step.

3.2.2. Hash-based generation of randomized interface identifiers

The algorithm in [[RFC7217](#)] can be augmented for the generation of temporary addresses. The benefit of this would be that a node could employ a single algorithm for generating stable and temporary addresses, by employing appropriate parameters.

Nodes would employ the following algorithm for generating the temporary IID:

1. Compute a random identifier with the expression:

```
RID = F(Prefix, MAC_Address, Network_ID, Time, DAD_Counter,  
secret_key)
```


Where:

RID:

Random Identifier

F():

A pseudorandom function (PRF) that MUST NOT be computable from the outside (without knowledge of the secret key). F() MUST also be difficult to reverse, such that it resists attempts to obtain the secret_key, even when given samples of the output of F() and knowledge or control of the other input parameters. F() SHOULD produce an output of at least 64 bits. F() could be implemented as a cryptographic hash of the concatenation of each of the function parameters. SHA-1 [[FIPS-SHS](#)] and SHA-256 are two possible options for F(). Note: MD5 [[RFC1321](#)] is considered unacceptable for F() [[RFC6151](#)].

Prefix:

The prefix to be used for SLAAC, as learned from an ICMPv6 Router Advertisement message.

MAC_Address:

The MAC address corresponding to the underlying network interface card. Employing the MAC address in this expression (in replacement of the Net_Iface parameter of the expression in [RFC7217](#)) means that the re-generation of a randomized MAC address will result in a different temporary address.

Network_ID:

Some network-specific data that identifies the subnet to which this interface is attached -- for example, the IEEE 802.11 Service Set Identifier (SSID) corresponding to the network to which this interface is associated. Additionally, Simple DNA [[RFC6059](#)] describes ideas that could be leveraged to generate a Network_ID parameter. This parameter is SHOULD be employed if some form of "Network_ID" is available.

Time:

An implementation-dependent representation of time. One possible example is the representation in UNIX-like systems [[OPEN-GROUP](#)], that measure time in terms of the number of seconds elapsed since the Epoch (00:00:00 Coordinated Universal Time (UTC), 1 January 1970).

DAD_Counter:

A counter that is employed to resolve Duplicate Address Detection (DAD) conflicts.

secret_key:

A secret key that is not known by the attacker. The secret key SHOULD be of at least 128 bits. It MUST be initialized to a pseudo-random number (see [RFC4086] for randomness requirements for security) when the operating system is installed or when the IPv6 protocol stack is "bootstrapped" for the first time.

2. The Interface Identifier is finally obtained by taking as many bits from the RID value (computed in the previous step) as necessary, starting from the least significant bit. The resulting Interface Identifier SHOULD be compared against the reserved IPv6 Interface Identifiers [RFC5453] [IANA-RESERVED-IID] and against those Interface Identifiers already employed in an address of the same network interface and the same network prefix. In the event that an unacceptable identifier has been generated, the value DAD_Counter should be incremented by 1, and the algorithm should be restarted from the first step.

3.3. Generating Temporary Addresses

[RFC4862] describes the steps for generating a link-local address when an interface becomes enabled as well as the steps for generating addresses for other scopes. This document extends [RFC4862] as follows. When processing a Router Advertisement with a Prefix Information option carrying a global scope prefix for the purposes of address autoconfiguration (i.e., the A bit is set), the node implementing this specification MUST perform the following steps:

1. Process the Prefix Information Option as defined in [RFC4862], either creating a new stable address or adjusting the lifetimes of existing addresses, both stable and temporary. If a received option will extend the lifetime of a stable address, the lifetimes of temporary addresses should be extended, subject to the overall constraint that no temporary addresses should ever remain "valid" or "preferred" for a time longer than (TEMP_VALID_LIFETIME - DESYNC_FACTOR) or (TEMP_PREFERRED_LIFETIME - DESYNC_FACTOR) respectively. The configuration variables TEMP_VALID_LIFETIME and TEMP_PREFERRED_LIFETIME correspond to approximate target lifetimes for temporary addresses.
2. One way an implementation can satisfy the above constraints is to associate with each temporary address a creation time (called CREATION_TIME) that indicates the time at which the address was created. When updating the preferred lifetime of an existing temporary address, it would be set to expire at whichever time is earlier: the time indicated by the received lifetime or (CREATION_TIME + TEMP_PREFERRED_LIFETIME - DESYNC_FACTOR). A

similar approach can be used with the valid lifetime.

3. When a new stable address is created as described in [[RFC4862](#)], or if the node has not configured any temporary address for the corresponding prefix, the node SHOULD create a new temporary address for such prefix.
4. When creating a temporary address, the lifetime values MUST be derived from the corresponding prefix as follows:
 - * Its Valid Lifetime is the lower of the Valid Lifetime of the stable address (if available) or TEMP_VALID_LIFETIME
 - * Its Preferred Lifetime is the lower of the Preferred Lifetime of the stable address (if available) or TEMP_PREFERRED_LIFETIME - DESYNC_FACTOR.
5. A temporary address is created only if this calculated Preferred Lifetime is greater than REGEN_ADVANCE time units. In particular, an implementation MUST NOT create a temporary address with a zero Preferred Lifetime.
6. New temporary addresses MUST be created by appending the interface's current randomized interface identifier to the prefix that was received.
7. The node MUST Perform duplicate address detection (DAD) on the generated temporary address. If DAD indicates the address is already in use, the node MUST generate a new randomized interface identifier, and repeat the previous steps as appropriate up to TEMP_IDGEN_RETRIES times. If after TEMP_IDGEN_RETRIES consecutive attempts no non-unique address was generated, the node MUST log a system error and MUST NOT attempt to generate temporary addresses for that interface. Note that DAD MUST be performed on every unicast address generated from this randomized interface identifier.

[3.4.](#) Expiration of Temporary Addresses

When a temporary address becomes deprecated, a new one MUST be generated. This is done by repeating the actions described in [Section 3.3](#), starting at step 3). Note that, except for the transient period when a temporary address is being regenerated, in normal operation at most one temporary address per prefix should be in a non-deprecated state at any given time on a given interface. Note that if a temporary address becomes deprecated as result of processing a Prefix Information Option with a zero Preferred Lifetime, then a new temporary address MUST NOT be generated. To

ensure that a preferred temporary address is always available, a new temporary address SHOULD be regenerated slightly before its predecessor is deprecated. This is to allow sufficient time to avoid race conditions in the case where generating a new temporary address is not instantaneous, such as when duplicate address detection must be run. The node SHOULD start the address regeneration process REGEN_ADVANCE time units before a temporary address would actually be deprecated.

As an optional optimization, an implementation MAY remove a deprecated temporary address that is not in use by applications or upper-layers as detailed in [Section 6](#).

3.5. Regeneration of Randomized Interface Identifiers

The frequency at which temporary addresses changes depends on how a device is being used (e.g., how frequently it initiates new communication) and the concerns of the end user. The most egregious privacy concerns appear to involve addresses used for long periods of time (weeks to months to years). The more frequently an address changes, the less feasible collecting or coordinating information keyed on interface identifiers becomes. Moreover, the cost of collecting information and attempting to correlate it based on interface identifiers will only be justified if enough addresses contain non-changing identifiers to make it worthwhile. Thus, having large numbers of clients change their address on a daily or weekly basis is likely to be sufficient to alleviate most privacy concerns.

There are also client costs associated with having a large number of addresses associated with a node (e.g., in doing address lookups, the need to join many multicast groups, etc.). Thus, changing addresses frequently (e.g., every few minutes) may have performance implications.

Nodes following this specification SHOULD generate new temporary addresses on a periodic basis. This can be achieved automatically by generating a new randomized interface identifier at least once every (TEMP_PREFERRED_LIFETIME - REGEN_ADVANCE - DESYNC_FACTOR) time units. As described above, generating a new temporary address REGEN_ADVANCE time units before a temporary address becomes deprecated produces addresses with a preferred lifetime no larger than TEMP_PREFERRED_LIFETIME. The value DESYNC_FACTOR is a random value (different for each client) that ensures that clients don't synchronize with each other and generate new addresses at exactly the same time. When the preferred lifetime expires, a new temporary address MUST be generated using the new randomized interface identifier.

Because the precise frequency at which it is appropriate to generate new addresses varies from one environment to another, implementations SHOULD provide end users with the ability to change the frequency at which addresses are regenerated. The default value is given in TEMP_PREFERRED_LIFETIME and is one day. In addition, the exact time at which to invalidate a temporary address depends on how applications are used by end users. Thus, the suggested default value of one week (TEMP_VALID_LIFETIME) may not be appropriate in all environments. Implementations SHOULD provide end users with the ability to override both of these default values.

Finally, when an interface connects to a new link, a new set of temporary addresses MUST be generated immediately. If a device moves from one ethernet to another, generating a new set of temporary addresses ensures that the device uses different randomized interface identifiers for the temporary addresses associated with the two links, making it more difficult to correlate addresses from the two different links as being from the same node. The node MAY follow any process available to it, to determine that the link change has occurred. One such process is described by Detecting Network Attachment [[DNA](#)].

3.6. Deployment Considerations

Devices implementing this specification MUST provide a way for the end user to explicitly enable or disable the use of temporary addresses. In addition, a site might wish to disable the use of temporary addresses in order to simplify network debugging and operations. Consequently, implementations SHOULD provide a way for trusted system administrators to enable or disable the use of temporary addresses.

Additionally, sites might wish to selectively enable or disable the use of temporary addresses for some prefixes. For example, a site might wish to disable temporary address generation for "Unique local" [[RFC4193](#)] prefixes while still generating temporary addresses for all other global prefixes. Another site might wish to enable temporary address generation only for the prefixes 2001::/16 and 2002::/16 while disabling it for all other prefixes. To support this behavior, implementations SHOULD provide a way to enable and disable generation of temporary addresses for specific prefix subranges. This per-prefix setting SHOULD override the global settings on the node with respect to the specified prefix subranges. Note that the pre-prefix setting can be applied at any granularity, and not necessarily on a per subnet basis.

The use of temporary addresses may cause unexpected difficulties with some applications. As described below, some servers refuse to accept

communications from clients for which they cannot map the IP address into a DNS name. In addition, some applications may not behave robustly if temporary addresses are used and an address expires before the application has terminated, or if it opens multiple sessions, but expects them to all use the same addresses.

If a very small number of nodes (say only one) use a given prefix for extended periods of time, just changing the interface identifier part of the address may not be sufficient to ensure privacy, since the prefix acts as a constant identifier. The procedures described in this document are most effective when the prefix is reasonably non static or is used by a fairly large number of nodes.

4. Implications of Changing Interface Identifiers

The desires of protecting individual privacy versus the desire to effectively maintain and debug a network can conflict with each other. Having clients use addresses that change over time will make it more difficult to track down and isolate operational problems. For example, when looking at packet traces, it could become more difficult to determine whether one is seeing behavior caused by a single errant machine, or by a number of them.

Some servers refuse to grant access to clients for which no DNS name exists. That is, they perform a DNS PTR query to determine the DNS name, and may then also perform an AAAA query on the returned name to verify that the returned DNS name maps back into the address being used. Consequently, clients not properly registered in the DNS may be unable to access some services. As noted earlier, however, a node's DNS name (if non-changing) serves as a constant identifier. The wide deployment of the extension described in this document could challenge the practice of inverse-DNS-based "authentication," which has little validity, though it is widely implemented. In order to meet server challenges, nodes could register temporary addresses in the DNS using random names (for example a string version of the random address itself).

Use of the extensions defined in this document may complicate debugging and other operational troubleshooting activities. Consequently, it may be site policy that temporary addresses should not be used. Consequently, implementations **MUST** provide a method for the end user or trusted administrator to override the use of temporary addresses.

5. Defined Constants

Constants defined in this document include:

TEMP_VALID_LIFETIME -- Default value: 1 week. Users should be able to override the default value.

TEMP_PREFERRED_LIFETIME -- Default value: 1 day. Users should be able to override the default value.

REGEN_ADVANCE -- 5 seconds

MAX_DESYNC_FACTOR -- 10 minutes. Upper bound on DESYNC_FACTOR.

DESYNC_FACTOR -- A random value within the range 0 - MAX_DESYNC_FACTOR. It is computed once at system start (rather than each time it is used) and must never be greater than (TEMP_VALID_LIFETIME - REGEN_ADVANCE).

TEMP_IDGEN_RETRIES -- Default value: 3

6. Future Work

An implementation might want to keep track of which addresses are being used by upper layers so as to be able to remove a deprecated temporary address from internal data structures once no upper layer protocols are using it (but not before). This is in contrast to current approaches where addresses are removed from an interface when they become invalid [[RFC4862](#)], independent of whether or not upper layer protocols are still using them. For TCP connections, such information is available in control blocks. For UDP-based applications, it may be the case that only the applications have knowledge about what addresses are actually in use. Consequently, an implementation generally will need to use heuristics in deciding when an address is no longer in use.

The determination as to whether to use stable versus temporary addresses can in some cases only be made by an application. For example, some applications may always want to use temporary addresses, while others may want to use them only in some circumstances or not at all. Suitable API extensions will likely need to be developed to enable individual applications to indicate with sufficient granularity their needs with regards to the use of temporary addresses. See [[I-D.gont-taps-address-usage-problem-statement](#)] for further details. Recommendations on DNS practices to avoid the problem described in [Section 4](#) when reverse DNS lookups fail may be needed. [[DNSOP](#)] contains a more detailed discussion of the DNS related issues.

While this document discusses ways of obscuring a user's IP address, the method described is believed to be ineffective against sophisticated forms of traffic analysis. To increase effectiveness, one may need to consider use of more advanced techniques, such as Onion Routing [[ONION](#)].

7. Security Considerations

Ingress filtering has been and is being deployed as a means of preventing the use of spoofed source addresses in Distributed Denial of Service (DDoS) attacks. In a network with a large number of nodes, new temporary addresses are created at a fairly high rate. This might make it difficult for ingress filtering mechanisms to distinguish between legitimately changing temporary addresses and spoofed source addresses, which are "in-prefix" (They use a topologically correct prefix and non-existent interface ID). This can be addressed by using access control mechanisms on a per address basis on the network egress point.

The security and privacy implications of IPv6 addresses are discussed in great detail in [[RFC7721](#)] and [[RFC7217](#)].

8. Significant Changes from RFC [RFC4941](#)

This section summarizes the changes in this document relative to [RFC 4941](#) that an implementer of [RFC 4941](#) should be aware of.

1. The algorithm to generate randomized interface identifiers was replaced by two possible alternative algorithms.
2. Generation of stable addresses is not implied or required by this document.
3. Temporary addresses are **not** disabled by default.

9. Acknowledgements

This document is based on [[RFC4941](#)] (authored by T. Narten, R. Draves, and S. Krishnan) and [[I-D.gont-6man-non-stable-ids](#)] (authored by F. Gont, C. Huitema, G. Gont, and M. Garcia Corbo).

10. References

10.1. Normative References

- [MD5] Rivest, R., "The MD5 Message-Digest Algorithm", [RFC 1321](#), April 1992.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [RFC 2119](#), March 1997.
- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", [BCP 106](#), [RFC 4086](#), DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/info/rfc4086>>.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", [RFC 4193](#), DOI 10.17487/RFC4193, October 2005, <<https://www.rfc-editor.org/info/rfc4193>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", [RFC 4291](#), DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", [RFC 4862](#), DOI 10.17487/RFC4862, September 2007, <<https://www.rfc-editor.org/info/rfc4862>>.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", [RFC 4941](#), DOI 10.17487/RFC4941, September 2007, <<https://www.rfc-editor.org/info/rfc4941>>.
- [RFC5453] Krishnan, S., "Reserved IPv6 Interface Identifiers", [RFC 5453](#), DOI 10.17487/RFC5453, February 2009, <<https://www.rfc-editor.org/info/rfc5453>>.
- [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", [RFC 6724](#), DOI 10.17487/RFC6724, September 2012, <<https://www.rfc-editor.org/info/rfc6724>>.
- [RFC7136] Carpenter, B. and S. Jiang, "Significance of IPv6 Interface Identifiers", [RFC 7136](#), DOI 10.17487/RFC7136, February 2014, <<https://www.rfc-editor.org/info/rfc7136>>.
- [RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address

Autoconfiguration (SLAAC)", [RFC 7217](#), DOI 10.17487/
[RFC7217](#), April 2014,
<<https://www.rfc-editor.org/info/rfc7217>>.

10.2. Informative References

- [COOKIES] Kristol, D. and L. Montulli, "HTTP State Management Mechanism", [RFC 2965](#), October 2000.
- [DDNS] Vixie, P., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", [RFC 2136](#), April 1997.
- [DHCP] Droms, R., "Dynamic Host Configuration Protocol", [RFC 2131](#), March 1997.
- [DHCPV6] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", [RFC 3315](#), July 2003.
- [DNA] Choi, J. and G. Daley, "Detecting Network Attachment in IPv6 Goals", [draft-ietf-dna-goals-04](#) (work in progress), December 2004.
- [DNSOP] Durand, A., Ihren, J., and P. Savola, "Operational Considerations and Issues with IPv6 DNS", [draft-ietf-dnsop-ipv6-dns-issues-10](#) (work in progress), October 2004.
- [FIPS-SHS] NIST, "Secure Hash Standard (SHS)", FIPS Publication 180-4, March 2012, <<http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>>.
- [I-D.gont-6man-non-stable-iids] Gont, F., Huitema, C., Gont, G., and M. Corbo, "Recommendation on Temporary IPv6 Interface Identifiers", [draft-gont-6man-non-stable-iids-01](#) (work in progress), March 2017.
- [I-D.gont-taps-address-usage-problem-statement] Gont, F., Gont, G., Corbo, M., and C. Huitema, "Problem Statement Regarding IPv6 Address Usage", [draft-gont-taps-address-usage-problem-statement-00](#) (work in progress), February 2018.
- [IANA-RESERVED-IIID] IANA, "Reserved IPv6 Interface Identifiers",

<<http://www.iana.org/assignments/ipv6-interface-ids>>.

[ONION] Reed, MGR., Syverson, PFS., and DMG. Goldschlag, "Proxies for Anonymous Routing", Proceedings of the 12th Annual Computer Security Applications Conference, San Diego, CA, December 1996.

[OPEN-GROUP]

The Open Group, "The Open Group Base Specifications Issue 7 / IEEE Std 1003.1-2008, 2016 Edition", [Section 4.16](#) Seconds Since the Epoch, 2016, <<http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/contents.html>>.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", [RFC 1321](#), DOI 10.17487/RFC1321, April 1992, <<https://www.rfc-editor.org/info/rfc1321>>.

[RFC6059] Krishnan, S. and G. Daley, "Simple Procedures for Detecting Network Attachment in IPv6", [RFC 6059](#), DOI 10.17487/RFC6059, November 2010, <<https://www.rfc-editor.org/info/rfc6059>>.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", [RFC 6151](#), DOI 10.17487/RFC6151, March 2011, <<https://www.rfc-editor.org/info/rfc6151>>.

[RFC7707] Gont, F. and T. Chown, "Network Reconnaissance in IPv6 Networks", [RFC 7707](#), DOI 10.17487/RFC7707, March 2016, <<https://www.rfc-editor.org/info/rfc7707>>.

[RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy Considerations for IPv6 Address Generation Mechanisms", [RFC 7721](#), DOI 10.17487/RFC7721, March 2016, <<https://www.rfc-editor.org/info/rfc7721>>.

Authors' Addresses

Fernando Gont
SI6 Networks / UTN-FRH
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires 1706
Argentina

Phone: +54 11 4650 8472
Email: fgont@si6networks.com
URI: <http://www.si6networks.com>

Suresh Krishnan
Ericsson Research
8400 Decarie Blvd.
Town of Mount Royal, QC
Canada

Email: suresh.krishnan@ericsson.com

