IPv6 maintenance Working Group (6man)
Internet-Draft
Intended status: Standards Track
Expires: December 5, 2013

F. Gont
SI6 Networks / UTN-FRH

June 3, 2013

A method for Generating Stable Privacy-Enhanced Addresses with IPv6 Stateless Address Autoconfiguration (SLAAC) draft-ietf-6man-stable-privacy-addresses-09

Abstract

This document specifies a method for generating IPv6 Interface Identifiers to be used with IPv6 Stateless Address Autoconfiguration (SLAAC), such that addresses configured using this method are stable within each subnet, but the Interface Identifier changes when hosts move from one network to another. This method is meant to be an alternative to generating Interface Identifiers based on hardware address (e.g., using IEEE identifiers), such that the benefits of stable addresses can be achieved without sacrificing the privacy of users. The method specified in this document applies to all prefixes a host may be employing, including link-local, global, and unique-local addresses.

Table of Contents

1. Introduction

[RFC4862] specifies Stateless Address Autoconfiguration (SLAAC) for IPv6 [RFC2460], which typically results in hosts configuring one or more "stable" addresses composed of a network prefix advertised by a local router, and an Interface Identifier (IID) that typically embeds a hardware address (e.g., using IEEE identifiers) [RFC4291].

> Cryptographically Generated Addresses (CGAs) [RFC3972] are yet another method for generating Interface Identifiers, which bind a public signature key to an IPv6 address in the SEcure Neighbor Discovery (SEND) [RFC3971] protocol.

Generally, the traditional SLAAC addresses are thought to simplify network management, since they simplify Access Control Lists (ACLs) and logging. However, they have a number of drawbacks:

- since the resulting Interface Identifiers do not vary over time, they allow correlation of node activities within the same network, thus negatively affecting the privacy of users.

- since the resulting Interface Identifiers are constant across networks, the resulting IPv6 addresses can be leveraged to track and correlate the activity of a node across multiple networks (e.g. track and correlate the activities of a typical client connecting to the public Internet from different locations), thus negatively affecting the privacy of users.

- since embedding the underlying link-layer address in the Interface Identifier will result in specific address patterns, such patterns may be leveraged by attackers to reduce the search space when performing address scanning attacks. For example, the IPv6 addresses of all nodes manufactured by the same vendor (at a given time frame) will likely contain the same IEEE Organizationally Unique Identifier (OUI) in the Interface Identifier.

- embedding the underlying link-layer address in the Interface Identifier leaks device-specific information that could be leveraged to launch device-specific attacks.

- embedding the underlying link-layer address in the Interface Identifier means that replacement of the underlying interface hardware will result in a change of the IPv6 address(es) assigned to that interface.

Appendix B provides additional details regarding how these vulnerabilities could be exploited, and the extent to which the method discussed in this document mitigates them.

The "Privacy Extensions for Stateless Address Autoconfiguration in IPv6" [RFC4941] (henceforth referred to as "temporary addresses") were introduced to complicate the task of eavesdroppers and other information collectors (e.g. IPv6 addresses in web server logs or email headers, etc.) to correlate the activities of a node, and basically result in temporary (and random) Interface Identifiers. These temporary addresses are generated in addition to the traditional IPv6 addresses based on IEEE identifiers, with the "temporary addresses" being employed for "outgoing communications", and the traditional SLAAC addresses being employed for "server" functions (i.e., receiving incoming connections).

It should be noted that temporary addresses can be challenging in a number of areas. For example, from a network-management point of view, they tend to increase the complexity of event logging, trouble-shooting, enforcement of access controls and quality of service, etc. As a result, some organizations disable the use of temporary addresses even at the expense of reduced privacy [Broersma]. Temporary addresses may also result in increased implementation complexity, which might not be possible or desirable in some implementations (e.g., some embedded devices).

In scenarios in which temporary addresses are deliberately not used (possibly for any of the aforementioned reasons), all a host is left with is the stable addresses that have been generated using e.g. IEEE identifiers. In such scenarios, it may still be desirable to have addresses that mitigate address scanning attacks, and that at the very least do not reveal the node's identity when roaming from one network to another -- without complicating the operation of the corresponding networks.

However, even with "temporary addresses" in place, a number of issues remain to be mitigated. Namely,

- since "temporary addresses" [RFC4941] do not eliminate the use of fixed identifiers for server-like functions, they only partially mitigate host-tracking and activity correlation across networks (see Appendix C.1 for some example attacks that are still possible with temporary addresses).

- since "temporary addresses" [RFC4941] do not replace the traditional SLAAC addresses, an attacker can still leverage patterns in those addresses to greatly reduce the search space for "alive" nodes [Gont-DEEPSEC2011] [CPNI-IPv6] [I-D.ietf-opsec-ipv6-host-scanning].

Hence, there is a motivation to improve the properties of "stable" addresses regardless of whether temporary addresses are employed or

not.

We note that attackers can employ a plethora of probing techniques [I-D.ietf-opsec-ipv6-host-scanning] to exploit the aforementioned issues. Some of them (such as the use of ICMPv6 Echo Request and ICMPv6 Echo Response packets) could mitigated by a personal firewall at the target host. For other vectors, such listening to ICMPv6 "Destination Unreachable, Address Unreachable" (Type 1, Code 3) error messages referring to the target addresses [I-D.ietf-opsec-ipv6-host-scanning], there is nothing a host can do (e.g., a personal firewall at the target host would not be able to mitigate this probing technique).

This document specifies a method to generate Interface Identifiers that are stable/constant for each network interface within each subnet, but that change as hosts move from one network to another, thus keeping the "stability" properties of the Interface Identifiers specified in [RFC4291], while still mitigating address-scanning attacks and preventing correlation of the activities of a node as it moves from one network to another.

The method specified in this document is a orthogonal to the use of "temporary" addresses [RFC4941], since it is meant to improve the security and privacy properties of the stable addresses that are employed along with the aforementioned "temporary" addresses. In scenarios in which "temporary addresses" are employed, implementation of the mechanism described in this document (in replacement of stable addresses based on e.g. IEEE identifiers) would mitigate address-scanning attacks and also mitigate the remaining vectors for correlating host activities based on the node's IPv6 addresses. On the other hand, for nodes that currently disable "temporary addresses" [RFC4941] for some of the reasons described earlier in this document, implementation of this mechanism will result in stable privacy-enhanced addresses which address some of the concerns related to addresses that embed IEEE identifiers [RFC4291], and which mitigate IPv6 address-scanning attacks.

We note that this method is incrementally deployable, since it does not pose any interoperability implications when deployed on networks where other nodes do not implement or employ it. Additionally, we note that this document does not update or modify IPv6 StateLess Address Auto-Configuration (SLAAC) [RFC4862] itself, but rather only specifies an alternative algorithm to generate Interface Identifiers. Therefore, the usual address lifetime properties (as specified in the corresponding Prefix Information Options) apply when IPv6 addresses are generated as a result of employing the algorithm specified in this document with SLAAC [RFC4862]. Additionally, from the point of view of renumbering, we note that these addresses behave like the

traditional IPv6 addresses (that embed a hardware address) resulting from SLAAC [RFC4862].

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 RFC 2119 [RFC2119].

2. Design goals

This document specifies a method for selecting Interface Identifiers to be used with IPv6 SLAAC, with the following goals:

- The resulting Interface Identifiers remain constant/stable for each prefix used with SLAAC within each subnet. That is, the algorithm generates the same Interface Identifier when configuring an address (for the same interface) belonging to the same prefix within the same subnet.

- The resulting Interface Identifiers do change when addresses are configured for different prefixes. That is, if different autoconfiguration prefixes are used to configure addresses for the same network interface card, the resulting Interface Identifiers must be (statistically) different. This means that, given two addresses produced by the method specified in this document, it must be difficult for an attacker tell whether the addresses have been generated/used by the same node.

- It must be difficult for an outsider to predict the Interface Identifiers that will be generated by the algorithm, even with knowledge of the Interface Identifiers generated for configuring other addresses.

- Depending on the specific implementation approach (see Section 3 and Appendix A), the resulting Interface Identifiers may be independent of the underlying hardware (e.g. link-layer address). This means that e.g. replacing a Network Interface Card (NIC) will not have the (generally undesirable) effect of changing the IPv6 addresses used for that network interface.

- The method specified in this document is meant to be an alternative to producing IPv6 addresses based on e.g. IEEE identifiers (as specified in [RFC2464]). It is meant to be employed for all of the stable (i.e. non-temporary) IPv6 addresses configured with SLAAC for a given interface, including global, link-local, and unique-local IPv6 addresses.

We note that of use of the algorithm specified in this document is (to a large extent) orthogonal to the use of "temporary addresses" [RFC4941]. When employed along with "temporary addresses", the method specified in this document will mitigate address-scanning attacks and correlation of node activities across networks (see Appendix C and [IAB-PRIVACY]). On the other hand, hosts that do not implement/use "temporary addresses" but employ the method specified in this document would, at the very least, mitigate the host-tracking and address scanning issues discussed in the previous section.

3. Algorithm specification

IPv6 implementations conforming to this specification MUST generate Interface Identifiers using the algorithm specified in this section in replacement of any other algorithms used for generating "stable" addresses (such as those specified in [RFC2464]). However, implementations conforming to this specification MAY employ the algorithm specified in [RFC4941] to generate temporary addresses in addition to the addresses generated with the algorithm specified in this document. The method specified in this document MUST be employed for generating the Interface Identifiers for all the stable addresses of a given interface, including IPv6 global, link-local, and unique-local addresses.

> This means that this document does not formally obsolete or deprecate any of the existing algorithms to generate Interface Identifiers (e.g. such as that specified in [RFC2464]). However, those IPv6 implementations that employ this specification MUST generate all of their "stable" addresses as specified in this document.

Implementations conforming to this specification SHOULD provide the means for a system administrator to enable or disable the use of this algorithm for generating Interface Identifiers.

Unless otherwise noted, all of the parameters included in the expression below MUST be included when generating an Interface Identifier.

1. Compute a random (but stable) identifier with the expression:

   RID = F(Prefix, Net_Iface, Network_ID, DAD_Counter, secret_key)

   Where:

   RID:
      Random (but stable) Interface Identifier

   F():
      A pseudorandom function (PRF) that is not computable from the outside (without knowledge of the secret key), which should produce an output of at least 64 bits.The PRF could be implemented as a cryptographic hash of the concatenation of each of the function parameters.

Prefix:
The prefix to be used for SLAAC, as learned from an ICMPv6 Router Advertisement message, or the link-local IPv6 unicast prefix.

Net_Iface:
An implementation-dependent stable identifier associated with the network interface for which the RID is being generated. An implementation MAY provide a configuration option to select the source of the identifier to be used for the Net_Iface parameter. A discussion of possible sources for this value (along with the corresponding trade-offs) can be found in Appendix A.

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. This parameter is OPTIONAL.

DAD_Counter:
A counter that is employed to resolve Duplicate Address Detection (DAD) conflicts. It MUST be initialized to 0, and incremented by 1 for each new tentative address that is configured as a result of a DAD conflict. Implementations that record DAD_Counter in non-volatile memory for each {Prefix, Net_Iface, Network_ID} tuple MUST initialize DAD_Counter to the recorded value if such an entry exists in non-volatile memory). See Section 4 for additional details.

secret_key:
A secret key that is not known by the attacker. The secret key MUST be initialized at operating system installation time to a pseudo-random number (see [RFC4086] for randomness requirements for security). An implementation MAY provide the means for the the system administrator to change or display the secret key.

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.

   We note that [RFC4291] requires that, the Interface IDs of all unicast addresses (except those that start with the binary value 000) be 64-bit 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).

The resulting Interface Identifier should be compared against the Subnet-Router Anycast [RFC4291] and the Reserved Subnet Anycast Addresses [RFC2526], 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, this situation should be handled in the same way as the case of duplicate addresses (see Section 4).

This document does not require the use of any specific PRF for the function F() above, since the choice of such PRF is usually a trade-off between a number of properties (processing requirements, ease of implementation, possible intellectual property rights, etc.), and since the best possible choice for F() might be different for different types of devices (e.g. embedded systems vs. regular servers) and might possibly change over time.

Note that the result of F() in the algorithm above is no more secure than the secret key. If an attacker is aware of the PRF that is being used by the victim (which we should expect), and the attacker can obtain enough material (i.e. addresses configured by the victim), the attacker may simply search the entire secret-key space to find matches. To protect against this, the secret key should be of a reasonable length. Key lengths of at least 128 bits should be adequate. The secret key is initialized at system installation time to a pseudo-random number, thus allowing this mechanism to be enabled/used automatically, without user intervention.

Including the SLAAC prefix in the PRF computation causes the Interface Identifier to vary across each prefix (link-local, global, etc.) employed by the node and, as consequently, also across networks. This mitigates the correlation of activities of multi-homed nodes (since each of the corresponding addresses will employ a different Interface ID), host-tracking (since the network prefix will change as the node moves from one network to another), and any other attacks that benefit from predictable Interface Identifiers (such as address scanning attacks).

The Net_Iface is a value that identifies the network interface for which an IPv6 address is being generated. The following properties are required for the Net_Iface parameter:

- it MUST be constant across system bootstrap sequences and other network events (e.g., bringing another interface up or down)

- it MUST be different for each network interface simultaneously in use

Since the stability of the addresses generated with this method relies on the stability of all arguments of F(), it is key that the Net_Iface be constant across system bootstrap sequences and other network events. Additionally, the Net_Iface must uniquely identify an interface within the node, such that two interfaces connecting to the same network do not result in duplicate addresses. Different types of operating systems might benefit from different stability properties of the Net_Iface parameter. For example, a client-oriented operating system might want to employ Net_Iface identifiers that are attached to the underlying network interface card (NIC), such that a removable NIC always gets the same IPv6 address, irrespective of the system communications port to which it is attached. On the other hand, a server-oriented operating system might prefer Net_Iface identifiers that are attached to system slots/ports, such that replacement of a network interface card does not result in an IPv6 address change. Appendix A discusses possible sources for the Net_Iface, along with their pros and cons.

Including the optional Network_ID parameter when computing the RID value above would cause the algorithm to produce a different Interface Identifier when connecting to different networks, even when configuring addresses belonging to the same prefix. This means that a host would employ a different Interface Identifier as it moves from one network to another even for IPv6 link-local addresses or Unique Local Addresses (ULAs). In those scenarios where the Network_ID is unknown to the attacker, including this parameter might help mitigate attacks where a victim node connects to the same subnet as the attacker, and the attacker tries to learn the Interface Identifier used by the victim node for a remote network (see Section 7 for further details).

The DAD_Counter parameter provides the means to intentionally cause this algorithm produce a different IPv6 addresses (all other parameters being the same). This could be necessary to resolve Duplicate Address Detection (DAD) conflicts, as discussed in detail in Section 4.

Finally, we note that all of the bits in the resulting Interface IDs are treated as "opaque" bits. For example, the universal/local bit of Modified EUI-64 format identifiers is treated as any other bit of such identifier. In theory, this might result in Duplicate Address Detection (DAD) failures that would otherwise not be encountered. However, this is not deemed as a real issue, because of the following considerations:

- The interface IDs of all addresses (except those of addresses that that start with the binary value 000) are 64-bit long. Since the method specified in this document results in random Interface IDs, the probability of DAD failures is very small.

- Real world data indicates that MAC address reuse is far more common than assumed [HDMoore]. This means that even IPv6 addresses that employ (allegedly) unique identifiers (such as IEEE identifiers) might result in DAD failures, and hence implementations should be prepared to gracefully handle such occurrences.

- Since some popular and widely-deployed operating systems (such as Microsoft Windows) do not employ unique hardware identifiers for the Interface IDs of their stable addresses, reliance on such unique identifiers is more reduced in the deployed world (fewer deployed systems rely on them for the avoidance of address collisions).

4. Resolving Duplicate Address Detection (DAD) conflicts

If as a result of performing Duplicate Address Detection (DAD) [RFC4862] a host finds that the tentative address generated with the algorithm specified in Section 3 is a duplicate address, it SHOULD resolve the address conflict by trying a new tentative address as follows:

- DAD_Counter is incremented by 1.
- A new Interface Identifier is generated with the algorithm specified in Section 3, using the incremented DAD_Counter value.

This procedure may be repeated a number of times until the address conflict is resolved. Hosts SHOULD try at least IDGEN_RETRIES (see Section 5) tentative addresses if DAD fails for successive generated addresses, in the hopes of resolving the address conflict. We also note that hosts MUST limit the number of tentative addresses that are tried (rather than indefinitely try a new tentative address until the conflict is resolved).

In those (unlikely) scenarios in which duplicate addresses are detected and in which the order in which the conflicting nodes configure their addresses may vary (e.g., because they may be bootstrapped in different order), the algorithm specified in this section for resolving DAD conflicts could lead to addresses that are not stable within the same subnet. In order to mitigate this potential problem, nodes MAY record the DAD_Counter value employed for a specific {Prefix, Net_Iface, Network_ID} tuple in non-volatile memory, such that the same DAD_Counter value is employed when configuring an address for the same Prefix and subnet at any other point in time.

In the event that a DAD conflict cannot be solved (possibly after trying a number of different addresses), address configuration would fail. In those scenarios, nodes MUST NOT automatically fall back to employing other algorithms for generating Interface Identifiers.

5. Specified Constants

This document specifies the following constant:

IDGEN_RETRIES:
defaults to 3.

6. IANA Considerations

There are no IANA registries within this document. The RFC-Editor can remove this section before publication of this document as an RFC.

7. Security Considerations

This document specifies an algorithm for generating Interface Identifiers to be used with IPv6 Stateless Address Autoconfiguration (SLAAC), as an alternative to e.g. Interface Identifiers that embed IEEE identifiers (such as those specified in [RFC2464]). When compared to such identifiers, the identifiers specified in this document have a number of advantages:

- They prevent trivial host-tracking, since when a host moves from one network to another the network prefix used for autoconfiguration and/or the Network ID (e.g., IEEE 802.11 SSID) will typically change, and hence the resulting Interface Identifier will also change (see Appendix C.1).

- They mitigate address-scanning techniques which leverage predictable Interface Identifiers (e.g., known Organizationally Unique Identifiers) [I-D.ietf-opsec-ipv6-host-scanning].

- They may result in IPv6 addresses that are independent of the underlying hardware (i.e. the resulting IPv6 addresses do not change if a network interface card is replaced) if an appropriate source for Net_Iface (Section 3) is employed.

- They prevent the information leakage produced by embedding hardware addresses in the Interface Identifier (which could be exploited to launch device-specific attacks).

- Since the method specified in this document will result in different Interface Identifiers for each configured address, knowledge/leakage of the Interface Identifier employed for one stable address of will not negatively affect the security/privacy of other stable addresses configured for other prefixes (whether at the same time or at some other point in time).

In scenarios in which an attacker can connect to the same subnet as a victim node, the attacker might be able to learn the Interface Identifier employed by the victim node for an arbitrary prefix, by simply sending a forged Router Advertisement [RFC4861] for that prefix, and subsequently learning the corresponding address configured by the victim node (either listening to the Duplicate Address Detection packets, or to any other traffic that employs the newly configured address). We note that a number of factors might limit the ability of an attacker to successfully perform such an attack:

- First-Hop security mechanisms such as RA-Guard [RFC6105] [I-D.ietf-v6ops-ra-guard-implementation] could prevent the forged

Router Advertisement from reaching the victim node

- If the victim implementation includes the (optional) Network_ID parameter for computing F() (see Section 3), and the Network_ID employed by the victim for a remote network is unknown to the attacker, the Interface Identifier learned by the attacker would differ from the one used by the victim when connecting to the legitimate network.

In any case, we note that at the point in which this kind of attack becomes a concern, a host should consider employing Secure Neighbor Discovery (SEND) [RFC3971] to prevent an attacker from illegitimately claiming authority for a network prefix.

We note that this algorithm is meant to be an alternative to Interface Identifiers such as those specified in [RFC2464], but is not meant as an alternative to temporary Interface Identifiers (such as those specified in [RFC4941]). Clearly, temporary addresses may help to mitigate the correlation of activities of a node within the same network, and may also reduce the attack exposure window (since temporary addresses are short-lived when compared to the addresses generated with the method specified in this document). We note that implementation of this algorithm would still benefit those hosts employing "temporary addresses", since it would mitigate host-tracking vectors still present when such addresses are used (see Appendix C.1), and would also mitigate address-scanning techniques that leverage patterns in IPv6 addresses that embed IEEE identifiers.

Finally, we note that the method described in this document addresses some of the privacy concerns arising from the use of IPv6 addresses that embed IEEE identifiers, without the use of temporary addresses, thus possibly offering an interesting trade-off for those scenarios in which the use of temporary addresses is not feasible.

8. Acknowledgements

The algorithm specified in this document has been inspired by Steven Bellovin's work ([RFC1948]) in the area of TCP sequence numbers.

The author would like to thank (in alphabetical order) Ran Atkinson, Karl Auer, Steven Bellovin, Matthias Bethke, Ben Campbell, Brian Carpenter, Tassos Chatzithomaoglou, Tim Chown, Alissa Cooper, Dominik Elsbroek, Brian Haberman, Bob Hinden, Christian Huitema, Ray Hunter, Jouni Korhonen, Eliot Lear, Jong-Hyouk Lee, Andrew McGregor, Tom Petch, Michael Richardson, Mark Smith, Dave Thaler, Ole Troan, James Woodyatt, and He Xuan, for providing valuable comments on earlier versions of this document.

This document is based on the technical report "Security Assessment of the Internet Protocol version 6 (IPv6)" [CPNI-IPv6] authored by Fernando Gont on behalf of the UK Centre for the Protection of National Infrastructure (CPNI).

Fernando Gont would like to thank CPNI (http://www.cpni.gov.uk) for their continued support.

9. References

9.1. Normative References

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

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

[RFC2526] Johnson, D. and S. Deering, "Reserved IPv6 Subnet Anycast Addresses", RFC 2526, March 1999.

[RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.

[RFC3542] Stevens, W., Thomas, M., Nordmark, E., and T. Jinmei, "Advanced Sockets Application Program Interface (API) for IPv6", RFC 3542, May 2003.

[RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.

[RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005.

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, July 2005.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, September 2007.

[RFC6105] Levy-Abegnoli, E., Van de Velde, G., Popoviciu, C., and J. Mohacsi, "IPv6 Router Advertisement Guard", RFC 6105, February 2011.

9.2. Informative References

[RFC1948] Bellovin, S., "Defending Against Sequence Number Attacks", RFC 1948, May 1996.

[RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, December 1998.

[I-D.ietf-opsec-ipv6-host-scanning] Gont, F. and T. Chown, "Network Reconnaissance in IPv6 Networks", draft-ietf-opsec-ipv6-host-scanning-01 (work in progress), April 2013.

[I-D.ietf-v6ops-ra-guard-implementation] Gont, F., "Implementation Advice for IPv6 Router Advertisement Guard (RA-Guard)", draft-ietf-v6ops-ra-guard-implementation-07 (work in progress), November 2012.

[HDMoore] HD Moore, "The Wild West", Louisville, Kentucky, U.S.A. September 25-29, 2012, <https://speakerdeck.com/hdm/derbycon-2012-the-wild-west>.

[Gont-DEEPSEC2011] Gont, "Results of a Security Assessment of the Internet Protocol version 6 (IPv6)", DEEPSEC 2011 Conference, Vienna, Austria, November 2011, <http://www.si6networks.com/presentations/deepsec2011/fgont-deepsec2011-ipv6-security.pdf>.

[Gont-BRUCON2012] Gont, "Recent Advances in IPv6 Security", BRUCON 2012 Conference, Ghent, Belgium, September 2012, <http://www.si6networks.com/presentations/brucon2012/fgont-brucon2012-recent-advances-in-ipv6-security.pdf>.

[Broersma] Broersma, R., "IPv6 Everywhere: Living with a Fully IPv6-enabled environment", Australian IPv6 Summit 2010, Melbourne, VIC Australia, October 2010, <http://www.ipv6.org.au/10ipv6summit/talks/Ron_Broersma.pdf>.

[IAB-PRIVACY] IAB, "Privacy and IPv6 Addresses", July 2011, <http://

www.iab.org/wp-content/IAB-uploads/2011/07/
IPv6-addresses-privacy-review.txt>.

[CPNI-IPv6]
Gont, F., "Security Assessment of the Internet Protocol version 6 (IPv6)", UK Centre for the Protection of National Infrastructure, (available on request).

Appendix A. Possible sources for the Net_Iface parameter

The following subsections describe a number of possible sources for the Net_Iface parameter employed by the F() function in Section 3. The choice of a specific source for this value represents a number of trade-offs, which may vary from one implementation to another.

A.1. Interface Index

The Interface Index [RFC3493] [RFC3542] of an interface uniquely identifies an interface within a node. However, these identifiers might or might not have the stability properties required for the Net_Iface value employed by this method. For example, the Interface Index might change upon removal or installation of a network interface (typically one with a smaller value for the Interface Index, when such a naming scheme is used), or when network interfaces happen to be initialized in a different order. We note that some implementations are known to provide configuration knobs to set the Interface Index for a given interface. Such configuration knobs could be employed to prevent the Interface Index from changing (e.g. as a result of the removal of a network interface).

A.2. Interface Name

The Interface Name (e.g., "eth0", "em0", etc) tends to be more stable than the underlying Interface Index, since such stability is required/desired when interface names are employed in network configuration (firewall rules, etc.). The stability properties of Interface Names depend on implementation details, such as what is the namespace used for Interface Names. For example, "generic" interface names such as "eth0" or "wlan0" will generally be invariant with respect to network interface card replacements. On the other hand, vendor-dependent interface names such as "rtk0" or the like will generally change when a network interface card is replaced with one from a different vendor.

We note that Interface Names might still change when network interfaces are added or removed once the system has been bootstrapped (for example, consider Universal Serial Bus-based network interface cards which might be added or removed once the system has been bootstrapped).

A.3. Link-layer Addresses

Link-layer addresses typically provide for unique identifiers for network interfaces; although, for obvious reasons, they generally change when a network interface card is replaced. In scenarios where neither Interface Indexes nor Interface Names have the stability

properties specified in Section 3 for Net_Iface, an implementation might want to employ the link-layer address of the interface for the Net_Iface parameter, albeit at the expense of making the corresponding IPv6 addresses dependent on the underlying network interface card (i.e., the corresponding IPv6 address would typically change upon replacement of the underlying network interface card).

A.4. Logical Network Service Identity

Host operating systems with a conception of logical network service identity, distinct from network interface identity or index, may keep a Universally Unique Identifier (UUID) [RFC4122] or similar identifier with the stability properties appropriate for use as the Net_Iface parameter.

Appendix B. Security/privacy issues with traditional SLAAC addresses

This section provides additional details regarding security/privacy issues arising from traditional SLAAC addresses- Namely, it provides additional details regarding how those issues could be exploited, and the extent to which the method specified in this document mitigates such issues.

B.1. Correlation of node activities within the same network

Since traditional SLAAC addresses employ Interface Identifiers that are constant within the same network, such identifiers can be leveraged to correlate the activities of a node within the same network. One sample scenario is that in which a client repeatedly connects to a server over a period of time, and hence, based on the stable Interface Identifier, the server can correlate all communication instances as being initiated by the same node.

The method specified in this document does not mitigate this attack vector, since it produces Interface Identifiers that are constant within a given network.

> This attack vector could only be mitigated by employing "temporary addresses" [RFC4941]. However, as noted earlier in this document, in scenarios in which there is a reduced number of nodes in a given network, mitigation of this vector might be difficult (if at all possible) -- even with "temporary addresses" [RFC4941] in place.

B.2. Correlation of node activities across networks

Since traditional SLAAC addresses employ Interface Identifiers that are constant across networks, such identifiers can be leveraged to correlate the activities of a node across networks. One sample scenario is that in which a client repeatedly connects to a server over a period of time, and hence, based on the stable Interface Identifier, the server can correlate all communication instances as being initiated by the same node. Since the method specified in this document results in Interface Identifiers that are not constant across networks, this attack vector is mitigated.

B.3. Host-tracking attacks

Since traditional SLAAC addresses employ Interface Identifiers that are constant across networks, such identifiers can be leveraged to track a node across networks.

For example, let us assume that the attacker knows the Interface

Identifier employed by the target node. If the target node contacts an attacker-operated node each time it moves from one network to another, the attacker will be able to track the node as it moves from one network to another.

An active version of this attack would imply that, once the Interface Identifier is known to the attacker the attacker probes whether there is an address with that Interface Identifier in each target network (i.e., in each network the client might connect to). If such address is found to be "alive", then the attacker could infer that the target node has connected to the corresponding network.

This vector is discussed in detail in Appendix C.1.2.

Since the method specified in this document results in Interface Identifiers that are not constant across networks, this attack vector is mitigated.

B.4. Address-scanning attacks

Since traditional SLAAC addresses typically embed the underlying link-layer address, the aforementioned addresses follow specific patterns that can be leveraged to reduce the search space when performing IPv6 address-scanning attacks (this is discussed in detail in [I-D.ietf-opsec-ipv6-host-scanning]). The method specified in this document produces random (but table within each subnet) Interface Identifiers, thus mitigating this attack vector.

B.5. Exploitation of device-specific information

Since traditional SLAAC addresses typically embed the underlying link-layer address, the aforementioned addresses leaks device-specific information that might be leveraged to launch device-specific attacks. For example, an attacker with knowledge about a specific vulnerability in devices manufactured by some vendor might easily identify potential targets by looking at the Interface Identifier of a list of IPv6 addresses. The method specified in this document produces random (but table within each subnet) Interface Identifiers, thus mitigating this attack vector.

Appendix C. Privacy issues still present when temporary addresses are employed

It is not unusual for people to assume or expect that all the security/privacy implications of traditional SLAAC addresses are mitigated when "temporary addresses" [RFC4941] are employed. However, as noted in Section 1 of this document and [IAB-PRIVACY], since temporary addresses are employed in addition to (rather than in replacement of) traditional SLAAC addresses, many of the security/privacy implications of traditional SLAAC addresses are not mitigated by the use of temporary addresses.

This section discusses a (non-exhaustive) number of scenarios in which host security/privacy is still negatively affected as a result of employing Interface Identifiers that are constant across networks (e.g., those resulting from embedding IEEE identifiers), even when temporary addresses [RFC4941] are employed. It aims to clarify the motivation of employing the method specified in this document in replacement of the traditional SLAAC addresses even when privacy/temporary addresses [RFC4941] are employed.

C.1. Host tracking

This section describes two attack scenarios which illustrate that host-tracking may still be possible when privacy/temporary addresses [RFC4941] are employed. These examples should remind us that one should not disclose more than it is really needed for achieving a specific goal (and an Interface Identifier that is constant across different networks does exactly that: it discloses more information than is needed for providing a stable address).

C.1.1. Tracking hosts across networks #1

A host configures its stable addresses with the constant Interface Identifier, and runs any application that needs to perform a server-like function (e.g. a peer-to-peer application). As a result of that, an attacker/user participating in the same application (e.g., P2P) would learn the constant Interface Identifier used by the host for that network interface.

Some time later, the same host moves to a completely different network, and employs the same P2P application. The attacker now interacts with the same host again, and hence can learn its newly-configured stable address. Since the Interface Identifier is the same as the one used before, the attacker can infer that it is communicating with the same device as before.

C.1.2. Tracking hosts across networks #2

Once an attacker learns the constant Interface Identifier employed by the victim host for its stable address, the attacker is able to "probe" a network for the presence of such host at any given network.

> See Appendix C.1.1 for just one example of how an attacker could learn such value. Other examples include being able to share the same network segment at some point in time (e.g. a conference network or any public network), etc.

For example, if an attacker learns that in one network the victim used the Interface Identifier 1111:2222:3333:4444 for its stable addresses, then he could subsequently probe for the presence of such device in the network 2011:db8::/64 by sending a probe packet (ICMPv6 Echo Request, or any other probe packet) to the address 2001:db8::1111:2222:3333:4444.

C.1.3. Revealing the identity of devices performing server-like functions

Some devices, such as storage devices, may typically perform server-like functions and may be usually moved from one network to another. Such devices are likely to simply disable (or not even implement) privacy/temporary addresses [RFC4941]. If the aforementioned devices employ Interface Identifiers that are constant across networks, it would be trivial for an attacker to tell whether the same device is being used across networks by simply looking at the Interface Identifier. Clearly, performing server-like functions should not imply that a device discloses its identity (i.e., that the attacker can tell whether it is the same device providing some function in two different networks, at two different points in time).

The scheme proposed in this document prevents such information leakage by causing nodes to generate different Interface Identifiers when moving from one network to another, thus mitigating this kind of privacy attack.

C.2. Address-scanning attacks

While it is usually assumed that IPv6 address-scanning attacks are unfeasible, an attacker can leverage address patterns in IPv6 addresses to greatly reduce the search space [I-D.ietf-opsec-ipv6-host-scanning] [Gont-BRUCON2012]. Addresses that embed IEEE identifiers result in one of such patterns that could be leveraged to reduce the search space when other nodes employ the same IEEE OUI (Organizationally Unique Identifier).

As noted earlier in this document, temporary addresses [RFC4941] do not replace/eliminate the use of IPv6 addresses that embed IEEE identifiers (they are employed in addition to those), and hence hosts implementing [RFC4941] would still be vulnerable to address-scanning attacks. The method specified in this document is meant as an alternative to addresses that embed IEEE identifiers, and hence eliminates such patterns (thus mitigating the aforementioned address-scanning attacks).

C.3. Information Leakage

IPv6 addresses embedding IEEE identifiers leak information about the device (Network Interface Card vendor, or even Operating System and/or software type), which could be leveraged by an attacker with device/software-specific vulnerabilities knowledge to quickly find possible targets. Since temporary addresses do not replace the traditional SLAAC addresses that typically embed IEEE identifiers, employing temporary addresses does not eliminate this possible information leakage.

C.4. Correlation of node activities within a network

In scenarios in which the number of nodes connected to a subnetwork is small, preventing the correlation of the activities of those nodes within such network might be difficult (if at all possible) to achieve, even with temporary addresses [RFC4941] in place. As a trivial example, consider a scenario where there is a single node (or a reduced number of nodes) connected to a specific network. An attacker could detect new addresses in use at that network along with addresses that are no longer in use, and infer which addresses are being employed by which hosts. This task is made particularly easier by the fact that use of "temporary addresses" can be easily inferred (since they follow different patterns from that of traditional SLAAC addresses), and since they are re-generated periodically (i.e., after a specific amount of time has elapsed).

Author's Address

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