IPv6 maintenance Working Group (6man)
Internet-Draft
Intended status: Informational
Expires: September 10, 2020

F. Gont
SI6 Networks / UTN-FRH
G. Gont
SI6 Networks
C. Huitema
Private Octopus Inc.

March 9, 2020

Problem Statement Regarding IP Address Usage
draft-gont-6man-address-usage-recommendations-05

Abstract

This document analyzes the security and privacy implications of IPv6 addresses based on a number of properties (such as address scope, stability, and usage type), and identifies gaps that currently prevent systems and applications from leveraging the increased flexibility and availability of IPv6 addresses.

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 https://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 10, 2020.

Table of Contents

1. Introduction

IPv6 addresses may differ in a number of properties, such as address scope (e.g. link-local vs. global), stability (e.g. stable addresses vs. temporary addresses), and intended usage type (outgoing communications vs. incomming communications). While often overlooked, these properties have impact on areas such as security, privacy, and performance.

IPv6 hosts typically configure a number of IPv6 addresses of different properties. For example, a host may configure one stable and one temporary address per each autoconfiguration prefix advertised on the local network. Currently, the addresses to be

configured typically depend on local system policy, with the aforementioned policy being static and irrespective of the network the host attaches to. This "one size fits all" approach limits the ability of systems and applications of fully-leveraging the increased flexibility and availability of IPv6 addresses.

Each application running on a given system may have its own set of requirements or expectations for the properties of the IPv6 addresses to be employed. For example, an application meaning to offer a public service might expect to employ global stable addresses for such purpose, while a privacy-sensible client application might prefer short-lived temporary addresses, or might even expect to employ single-use ("throw-away") IPv6 addresses when connecting to public servers. However, the subtetlies associated with IPv6 addresses (and associated properties) are often ignored by application programmers and, in any case, current APIs (such as the BSD Sockets API) tend to be very limited in the amount of control they give applications to select the most appropriate IPv6 addresses for a given task, thus limiting a programmer's ability to leverage IPv6 address availability and properties.

This document analyzes the impact of a number of properties of IPv6 addresses on areas such as security and privacy, and analyzes how IPv6 addresses are curently generated and employed by different operating systems and applications. Finally, it provides a problem statement by identifying and analyzing gaps that prevent systems and applications from fully-leveraging IPv6 addressing capabilities, setting the basis for new work that could fill those gaps.

2. Terminology

This document employs the definitions of "public address", "stable address", and "temporary address" from Section 2 of [RFC7721].

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].

3. Background

Predictable IPv6 addresses result in a number of security and privacy implications. For example, [Barnes2012] discusses how patterns in network prefixes can be leveraged for IPv6 address scanning. On the other hand, [RFC7707], [RFC7721] and [RFC7217] discuss the security and privacy implications of predictable IPv6 Interface Identifiers (IIDs).

Given the aforementioned previous work in this area, and the formal specification update produced by [RFC8064], we expect (and assume in the rest of this document) that implementations have replaced any schemes that produce predictable addresses with alternative schemes that avoid such patterns (e.g., RFC7217 in replacement of the traditional SLAAC addresses that embed link-layer addresses).

4. IPv6 Address Properties

There are three parameters that affect the security and privacy properties of an IPv6 address:

- Scope
- Stability
- Usage type (client-like "outgoing connections" vs. server-like "incoming connections")

Section 4.1, Section 4.2, and Section 4.3 discuss the security and privacy implications (and associated tradeoffs) of the scope, stability and usage type properties of IPv6 addresses, respectively.

Additionally, IPv6 address usage has a number of operational considerations; these are discussed n Section 7.5.

4.1. Address Scope Considerations

The IPv6 address scope can, in some scenarios, limit the attack exposure of a node as a result of the implicit isolation provided by a non-global address scope. For example, a node that only employs link-local addresses may, in principle, only be exposed to attack from other nodes in the local link. Hosts employing only Unique Local Addresses (ULAs) may be more isolated from attack than those employing Global Unicast Addresses (GUAs), assuming that proper packet filtering is enforced at the network edge.

The potential protection provided by a non-global addresses should not be regarded as a complete security strategy, but rather as a form of "prophylactic" security (see [I-D.gont-opsawg-firewalls-analysis]).

We note that the use of non-global addresses is usually limited to a reduced type of applications/protocols that e.g. are only meant to operate on a reduced scope, and hence their applicability may be limited.

A discussion of ULA usage considerations can be found in [I-D.ietf-v6ops-ula-usage-considerations].

4.2. Address Stability Considerations

The stability of an address has two associated security/privacy implications:

- Ability of an attacker to correlate network activity
- Exposure to attack

For obvious reasons, an address that is employed for multiple communication instances allows the aforementioned network activities to be correlated. The longer an address is employed (i.e., the more stable it is), the longer such correlation will be possible. In the worst-case scenario, a stable address that is employed for multiple communication instances over time will allow all such activities to be correlated. On the other hand, if a host were to generate (and eventually "throw away") one new address for each communication instance (e.g., TCP connection), network activity correlation would be mitigated.

NOTE:
The use of constant IIDs (as in traditional SLAAC) result in addresses that, while not constant as a whole (since the prefix changes), contain a globally-unique value that leaks out the node "identity". Such addresses result in the worst possible security and privacy implications, and their use has been deprecated by [RFC8064].

Typically, when it comes to attack exposure, the longer an address is employed the longer an attacker is exposed to attacks (e.g. an attacker has more time to find the address in the first place [RFC7707]). While such exposure is traditionally associated with the stability of the address, the usage type of the address (see Section 4.3) may also have an impact on attack exposure.

A popular approach to mitigate network activity correlation is the use of "temporary addresses" [RFC4941]. Temporary addresses are typically configured and employed along with stable addresses, with the temporary addresses employed for outgoing communications, and the stable addresses employed for incoming communications.

NOTE:
Ongoing work [I-D.gont-6man-non-stable-iids] aims at updating [RFC4941] such that temporary addresses can be employed without the need to configure stable addresses.

We note that the extent to which temporary addresses provide improved mitigation of network activity correlation and/or reduced attack exposure may be questionable and/or limited in some scenarios. For example, a temporary address that is reachable for, say, a few hours has a questionable "reduced exposure" (particularly when automated attack tools do not typically require such a long period of time to complete their task). Similarly, if network activity can be correlated for the life of such address (e.g., on the order of several hours), such period of time might be long enough for the attacker to correlate all the network activity he is meaning to correlate.

In order to better mitigate network activity correlation and/or possibly reduce host exposure, an implementation might want to either reduce the preferred lifetime of a temporary address, or even better, generate one new temporary address for each new transport protocol instance. However, the associated lifetime/stability of an address may have a negative impact on the network. For example, if a node were to employ "throw away" IPv6 addresses, or employ temporary addresses [RFC4941] with a short preferred lifetime, local nodes might need to maintain too many entries in their Neighbor Cache, and a number of devices (possibly enforcing security policies) might also need to cope with such additional state.

Additionally, enforcing a maximum lifetime on IPv6 addresses may cause long-lived TCP connections to fail. For example, an address becoming "Invalid" (after transitioning through the "Preferred" and "Deprecated" states) would cause the TCP connections employing them to break. This, in turn, would cause e.g. long-lived SSH sessions to break/fail.

In some scenarios, attack exposure may be reduced by limiting the usage of temporary addresses to outgoing connections, and prevent such addresses from being used for incoming connections (please see Section 4.3).

4.3. Usage Type Considerations

A node that employs one of its addresses to communicate with an external server (i.e., to perform an "outgoing connection") may cause such address to become exposed to attack. For example, once the external server receives an incoming connection, the corresponding server might launch an attack against the aforementioned address. A real-world instance of this type of scenario has been documented in [Hein].

However, we note that employing an IPv6 address for outgoing communications need not increase the exposure of local services to

other parties. For example, nodes could employ temporary addresses only for outgoing connections, but not for incoming connections. Thus, external nodes that learn about client's addresses could not really leverage such addresses for actively contacting the clients.

There are multiple ways in which this could possibly be achieved, with different implications. Namely:

- Run a host-based or network-based firewall
- Bind services to specific (explicit) addresses
- Bind services only to stable addresses

A client could simply run a host-based firewall that only allows incoming connections on the stable addresses. This is clearly more of an operational way of achieving the desired functionality, and may require good firewall/host integration (e.g., the firewall should be able to tell stable vs. temporary addresses), may require the client to run additional firewall software for this specific purpose, etc. In other scenarios, a network-based firewall could be configured to allow outgoing communications from all internal addresses, but only allow incoming communications to stable addresses. For obvious reasons, this is generally only applicable to networks where incoming communications are allowed to a limited number of hosts/servers.

Services could be bound to specific (explicit) addresses, rather than to all locally-configured addresses. However, there are a number of short-comings associated with this approach. Firstly, an application would need to be able to learn all of its addresses and associated stability properties, something that tends to be non-trivial and non-portable, and that also makes applications protocol-dependent, unnecessarily. Secondly, the BSD Sockets API does not really allow a socket to be bound to a subset of the node's addresses. That is, sockets can be bound to a single address or to all available addresses (wildcard), but not to a subset of all the configured addresses.

Binding services only to stable addresses provides a clean separation between addresses employed for client-like outgoing connections and server-like incoming connections. However, we currently lack an appropriate API for nodes to be able to specify that a socket should only be bound to stable addresses.

5. Default Address Selection in IPv6

Applications use system API's to select the IPv6 addresses that will be used for incoming and outgoing connections. These choices have consequences in terms of privacy, security, stability and performance.

Default Address Selection for IPv6 is specified in [RFC6724]. The selection starts with a set of potential destination addresses, such as returned by getaddrinfo(), and the set of potential source addresses currently configured for the selected interfaces. For each potential destination address, the algorithm will select the source address that provides the best route to the destination, while choosing the appropriate scope and preferring temporary addresses. The algorithm will then select the destination address, while giving a preference to reachable addresses with the smallest scope. The selection may be affected by system settings. We note that [RFC6724] only applies for outgoing connections, such as those made by clients trying to use services offered by other hosts.

We note that [RFC6724] selects IPv6 addresses from all the currently available addresses on the host, and there is currently no way for an application to indicate expected or desirable properties for the IPv6 source addresses employed for such outgoing communications. For example, a privacy-sensitive application might want that each outgoing communication instance employs a new, single-use IPv6 address, or to employ a new reusable address that is not employed or reusable by any other application on the host. Reuse of an IPv6 address by an application would allow the correlation of all network activities corresponding to such application as being performed by the same host, while reuse of an IPv6 address by multiple different applications would allow the correlation of all such network activities as being performed by the host with such IPv6 address.

When devices provide a service, the common pattern is to just wait for connections over all addresses configured on the device. For example, applications using the BSD Sockets API will commonly bind() the listening socket to the undefined address. This long-established behavior is appropriate for devices providing public services, but may have unexpected results for devices providing semi-private services, such as various forms of peer-to-peer or local-only applications.

This behavior leads to three problems: device tracking, discussed in Section 7.1.2; unexpected address discovery, discussed in Section 7.1.3; and availability outside the expected scope, discussed in Section 7.1.4. These problems are caused in part by the

limitations of available address selection API, presented in Section 7.2.

6. Current Possible Approaches for IPv6 Address Usage

6.1. Incoming communications

There are a number of ways in which a system or network may affect which address (and how) may be employed for different services and cases. Namely,

- TCP/IP stack address filtering
- Application-based address filtering
- Firewall-based address filtering

Clearly, the most elegant approach for address selection is for applications to be able to specify the properties of the addresses they are willing to employ by means of an API, such the TCP/IP stack itself can "filter" which addresses are allowed to be employed for the given service/application. This relieves the application from dealing with low level details of networking, improves portability, and avoids duplicate code in applications. However, constraints in the current APIs (see Section 7.2) may limit the ability of application progremmers for leveraging this technique.

Another possible approach is for applications to e.g. bind services to all available addresses, and perform the associated selection/ filtering at the application level. While possible this has a number of drawbacks. Firstly, it would require applications to deal with low-level networking details, require that all the associated code be duplicated in all applications, and also negatively affect portability. Besides, performing address/selection filtering at the application level may not mitigate some possible threats. For example, port scanning will still be possible, since the aforementioned filtering will only be performed e.g. once UDP packets are received or TCP connections are established.

Finally, a firewall may be employed to filter addresses based on their intended usage. For example, a firewall may block incoming requests to all addresses except to some whitelisted addresses (such as the stable addresses of the node). This technique not only requires the use of a firewall (which may or may not be present), but also implies knowledge of the firewall regarding the desired properties of the addresses that each application/service is intended to use.

6.2. Outgoing communications

An application might be able to obtain the list of currently-configured addresses, and subsequently select an address with desired properties, and explicitly "bind" the address to the socket, to override the default source address selection.

However, this approach is problematic for a number of reasons. Firstly, there is no portable way of obtaining the list of currently-configured addresses on the local node, and even less to check for properties such "valid lifetime". Secondly, as discussed in Section 6.1, it would require application programmers to understand all the subtetiles associated with IPv6 addressing, and would also lead to duplicate code on all applications. Finally, applications would be limited to use already-configured addresses and unable to trigger the generation of new addresses where desirable (e.g. the genration of a new temporary address for this application instance or communication instance).

7. Problem Statement

This section elaborates the problem statement on IPv6 address usage. Section 7.1 describes the security and privacy implications of improper IPv6 address usage, while Section 7.2, Section 7.4, Section 7.3, analyze the possible root of such improper address usage, suggesting possible future work.

7.1. Issues Associated with Sub-optimal IPv6 Address Usage

7.1.1. Correlation of Network Activity

As discussed in [RFC7721], a node that reuses an IPv6 address for multiple communication instances would allow the correlation of such network activities. This could be the case when the same IPv6 address is employed by several instances of the same application (e.g., a browser in "privacy" mode and a browser in "normal" mode), or when the same IPv6 address is employed by two different applications on the same node (e.g., a browser in "privacy" mode, and an email client).

Particularly for privacy-sensitive applications, an application or system might want to limit the usage of a given IPv6 address to a single communication instance, a single application, a single user on the system, etc. However, given current APIs, this is practically impossible.

7.1.2. Testing for the Presence of Node in the Network

The stable addresses recommended in [RFC8064] use stable IIDs defined in [RFC7217]. One key part of that algorithm is that if a device connects to a given network at different times, it will always configure the same IPv6 addresses on that network. If the device hosts a service ready to accept connections on that stable address, adversaries can test the presence of the device on the network by attempting connections to that stable address. Stable addresses used by listening services will thus enable testing whether a specific device is returning to a particular network, which in a number of cases might be considered a privacy issue.

7.1.3. Unexpected Address Discovery

Systems like DNS-Based Service Discovery [RFC6763] allow clients to discover services within a limited scope, that can be defined by a domain name. These services are not advertised outside of that scope, and thus do not expect to be discovered by random parties on the Internet. However, such services may be easily discoverable if they listen for connections to IPv6 addresses that a client process also uses as source address when connecting to remote servers.

NOTE:
An example of such unexpected discovery is described in [Hein]. A network manager observed scanning traffic directed at the temporary addresses of local devices. The analysis in [Hein] shows that the scanners learned the addresses by observing the device contact an NTP service ([RFC5905]). The remote scanning was possible because the local devices were also accepting connections directed to the temporary addresses.

It is obvious from the example that the "attack surface" of the services is increased because they are bond to the same IPv6 addresses that are also used by clients for outgoing communications with remote systems. But the overlap between "client" and "server" addresses is only one part of the problem. Suppose that a device hosts both a video game and a home automation application. The video game users will be able to discover the IPv6 address of the game server. If the home automation server listens to the same IPv6 addresses, it is now exposed to connection attempts by all these users. That, too, increases the attack surface of the home automation server.

7.1.4. Availability Outside the Expected Scope

The IPv6 addressing architecture [RFC4291] defines multiple address scopes. In practice, devices are often configured with globally reachable unicast addresses, link local addresses, and Unique Local IPv6 Unicast Addresses (ULA) [RFC4193]. Availability outside the expected scope happens when a service is expected to be only available in some local scope, but inadvertently becomes available to remote parties. That could happen for example if a service is meant to be available only on a given link, but becomes reachable through ULA or through globally reachable addresses, or if a service is meant to be available only inside some organization's perimeter and becomes reachable through globally reachable addresses. It will happen in particular if a service intended for some local scope is programmed to bind to "unspecified" addresses, which in practice means every address configured for the device (please see Section 7.2).

7.2. Current Limitations in the Address Selection APIs

Application developers using the BSD Sockets API can "bind" a listening socket to a specific address, and ensure that the application is only reachable through that address. In theory, careful selection of the binding address could mitigate the problems described in Section 7.1. Binding services to temporary addresses could mitigate the ability of an attacker from testing for the presence of the node in the network. Binding different services to different addresses could mitigate unexpected discovery. Binding services to link local addresses or ULA could mitigate availability outside the expected scope. However, explicitly managing addresses adds significant complexity to the application development. It requires that application developers master addressing architecture subtleties, and implement logic that reacts adequately to connectivity events and address changes. Experience shows that application developers would probably prefer some much simpler solution.

In addition, we should note that many application developers use high level APIs that listen to TLS, HTTP, or some other application protocol. These high level APIs seldom provide detailed access to specific IP addresses, and typically default to listening to all available addresses.

A more advanced API could allow an application programmer to select desired properties in an address (scope, lifespan, etc.), such that the best-suitable addresses are selected, while relieving the application for low-level IPv6 addressing details. Such API might also trigger the generation of new IPv6 addresses when the specified properties would require so.

7.3. Sub-optimal IPv6 Address Configuration

Most operating systems configure the same types of addresses regardless of the current "operating mode" or "profile" of the device (e.g., device connected to enterprise network vs roaming across untrusted networks). For example, many operating systems configure both stable [RFC8064] and temporary [RFC4941] addresses on all network interfaces. However, this "one size fits all" approach tends to be sub-optimal or inappropriate for some scenarios. For example, enterprise networks typically prefer usage of only stable address, thus meaning that a network administator needs to find the means for disabling the generation of temporary addresses on all those systems that would otherwise generate them. On the other hand, some mobile devices configure both stable and temporary addresses, even when their usage pattern (client-like operation, as opposed to offering services to other nodes) would allow for the more privacy-sensible option of configuring only temporary addresses.

The lack of better tuned address configuration policies has helped the "one size fits all" approach that, as noted, may lead to suboptimal results. Advice in this area might help achieve more optional address generation policies such that IPv6 addressing capabilities are fully leveraged.

NOTE:
One might envision a document that provides advice regarding the address generation for different typical scenarios (e.g., when to configure stable-only, temporary-only, or stable+temporary). In the most simple analysis, one might expect nodes in a typical enterprise network to employ only stable addresses. General-purpose nodes in a home or "trusted" network may want to employ both stable and temporary addresses. Finally, mobile nodes (e.g. when roaming across non-trusted networks) may want to employ only temporary addresses).

7.4. Sub-optimal IPv6 Address Usage

An application programmer, left with the question of which are the most appropriate addresses for a given usage type and application, typically resorts to the Default IPv6 Address Selection for IPv6 (see Section 5) for outgoing communications, and to accepting incoming communications on all available addresses for incoming communications. As discussed throughout this document, this leads to sub-optimal results. Besides, all applications on a node share the same pool of configured addresses, and applications are also prevented from triggering the generation of new addresses (e.g. to be employed for a particular application or communcation instance).

Guidance in this area is warranted such that applications and systems fully-leverage IPv6 addressing.

NOTE:
Such guidance would elaborate, among other things, on the usage of IPv6 addresses when offering network services and when performing client-like communications. For example, for incomming communications, hosts might want to employ only the smallest-scope applicable addresses (if available) and, if stable addresses are available, they might want to accept incoming connections only on such addresses (but *not* on temporary addresses). For client-like communications, hosts might prefer temporary addresses, unless the coresponding communication instances are expected to be long-lived (e.g., SSH sessions).

7.5. Operational Considerations

The desires of protecting individual privacy versus the desire to effectively maintain and debug a network can conflict with each other. Having clients e.g. 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.

Network deployments are currently recommended to provide multiple IPv6 addresses from each prefix to general-purpose hosts [RFC7934]. However, in some scenarios, use of a large number of IPv6 addresses may have negative implications on network devices that need to maintain entries for each IPv6 address in some data structures (e.g., [RFC7039]). Additionally, concurrent active use of multiple IPv6 addresses will increase neighbour discovery traffic if Neighbour Caches in network devices are not large enough to store all addresses on the link. This can impact performance and energy efficiency on networks on which multicast is expensive (e.g. [I-D.ietf-mboned-ieee802-mcast-problems]).

8. 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.

9. Security Considerations

The security and privacy implications associated with the predictability and lifetime of IPv6 addresses has been analyzed in [RFC7217] [RFC7721], and [RFC7707]. This document complements and

extends the aforementioned analysis by considering other IPv6 properties such as the address scope and address usage type, and the associated tradeoffs. Finally, it describes possible future standards-track work to allow for greater flexibility in IPv6 address usage.

10. Acknowledgements

The authors would like to thank (in alphabetical order) Francis Dupont, Tatuya Jinmei, and Dave Thaler for providing valuable comments on earlier versions of this document.

Section 7.5 borrows text from [I-D.ietf-6man-rfc4941bis] co-authored by Fernando Gont, Suresh Krishnan, Thomas Narten, and Richard Draves.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[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>.

[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>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

[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>.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <https://www.rfc-editor.org/info/rfc6763>.

[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>.

[RFC7934] Colitti, L., Cerf, V., Cheshire, S., and D. Schinazi, "Host Address Availability Recommendations", BCP 204, RFC 7934, DOI 10.17487/RFC7934, July 2016, <https://www.rfc-editor.org/info/rfc7934>.

[RFC8064] Gont, F., Cooper, A., Thaler, D., and W. Liu, "Recommendation on Stable IPv6 Interface Identifiers", RFC 8064, DOI 10.17487/RFC8064, February 2017, <https://www.rfc-editor.org/info/rfc8064>.

11.2. Informative References

[Barnes2012] Barnes, R., Altmann, R., and D. Kerr, "Mapping the Great Void Smarter scanning for IPv6", ISMA 2012 AIMS-4 - Workshop on Active Internet Measurements, February 2012, <https://www.caida.org/workshops/isma/1202/slides/aims1202_rbarnes.pdf>.

[Hein] Hein, B., "The Rising Sophistication of Network Scanning", January 2016, <http://netpatterns.blogspot.be/2016/01/the-rising-sophistication-of-network.html>.

[I-D.gont-6man-non-stable-iids] Gont, F., Huitema, C., Krishnan, S., Gont, G., and M. Corbo, "Recommendation on Temporary IPv6 Interface Identifiers", draft-gont-6man-non-stable-iids-04 (work in progress), March 2018.

[I-D.gont-opsawg-firewalls-analysis] Gont, F. and F. Baker, "On Firewalls in Network Security", draft-gont-opsawg-firewalls-analysis-02 (work in progress), February 2016.

[I-D.ietf-6man-rfc4941bis]
Gont, F., Krishnan, S., Narten, T., and R. Draves, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", draft-ietf-6man-rfc4941bis-07 (work in progress), February 2020.

[I-D.ietf-mboned-ieee802-mcast-problems]
Perkins, C., McBride, M., Stanley, D., Kumari, W., and J. Zuniga, "Multicast Considerations over IEEE 802 Wireless Media", draft-ietf-mboned-ieee802-mcast-problems-11 (work in progress), December 2019.

[I-D.ietf-v6ops-ula-usage-considerations]
Liu, B. and S. Jiang, "Considerations For Using Unique Local Addresses", draft-ietf-v6ops-ula-usage-considerations-02 (work in progress), March 2017.

[RFC7039] Wu, J., Bi, J., Bagnulo, M., Baker, F., and C. Vogt, Ed., "Source Address Validation Improvement (SAVI) Framework", RFC 7039, DOI 10.17487/RFC7039, October 2013, <https://www.rfc-editor.org/info/rfc7039>.

[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

Guillermo Gont
SI6 Networks
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires 1706
Argentina

Phone: +54 11 4650 8472
Email: ggont@si6networks.com
URI: https://www.si6networks.com

Christian Huitema
Private Octopus Inc.
Friday Harbor, WA 98250
U.S.A.

Email: huitema@huitema.net
URI: http://privateoctopus.com