IPv6 Address Usage Recommendations
draft-gont-6man-address-usage-recommendations-03

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. It analyzes what properties are desirable
   for some popular scenarios, and provides advice regarding the
   configuration and usage of such addresses.

Table of Contents

1. Introduction

IPv6 hosts typically configure a number of IPv6 addresses, which may
differ in multiple aspects, such as address scope and stability (e.g.
stable addresses vs. temporary addresses). For example, a host may
configure one stable and one temporary address per each
autoconfiguration prefix advertised on the local network. The
addresses to be configured typically depend on local system policy
configuration, with the aforementioned policy being static and
irrespective of the network the host attaches to.

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

o Scope

   o Stability

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

   Section 5.1, Section 5.2, and Section 5.3 discuss the security and
   privacy implications (and associated tradeoffs) of the scope,
   stability and usage type properties of IPv6 addresses, respectively.

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. Problem Statement

   Applications use system API's to select the IPv6 addresses that will
   be used for incoming and outgoing connections. This 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.
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 Sockets API will commonly bind() the
listening socket to 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 4.1.1; unexpected address discovery, discussed in
Section 4.1.2; and availability outside the expected scope, discussed
in Section 4.1.3. These problems are caused in part by the
limitations of available address selection API, presented in
Section 4.2.

4.1. Issues Associated with Sub-optimal IPv6 Address Usage

4.1.1. Testing for the Presence of Node in the Network

The stable addresses recommended in [RFC8064] use stable IID 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 will be considered a privacy issue.

4.1.2. 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. Yet it appears that such services are 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. Analysis 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 devices
   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.

4.1.3. 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.

4.2. Current Limitations in the Address Selection API

   Application developers using the 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 three problems mentioned above.
   Binding services to temporary address could mitigate device tracking.
   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 architectures 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.

[5](#). IPv6 Address Considerations

[5.1](#). 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](#)].

[5.2](#). Address Stability Considerations

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

o Ability of an attacker to correlate network activity

o 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 5.3) 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 keep 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 outbound connections, and prevent
such addresses from being used for inbound connections (please see
Section 5.3).

5.3. 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 an 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:

o Run a host-based or network-based firewall

o Bind services to specific (explicit) addresses

o 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 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. Development of such an API should
be considered for future work.

6. Possible Approaches for IPv6 Address Usage

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,

o TCP/IP stack address filtering

o Application-based address filtering

o 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 4.2) may limit the ability of
application progremmers form 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.

7. Advice on IPv6 Address Configuration

[TBD]

TODO: This section is expected to provide advice regarding the
configuration of different addresses for different typical scenarios.
e.g., when nodes may want 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).

8. Advice on IPv6 Address Usage

[TBD]

TODO: This section is expected to provide recommendations regarding
the usage of IPv6 addresses. Among others, it is expected to provide

recommendations regarding the usage of IPv6 addresses when providing network services. In the mos simple form, one argue that nodes may want to employ only the smallest-scope applicable addresses (if available) and, if stable addresses are available, nodes may want to accept incoming connections on such addresses but *not* on temporary addresses.

9. Future Work

Some of the discussion in this document suggest that in order to fully benefit from the IPv6 addresses (in terms of e.g. increased availability of addresses and address types) additional work may be required in this areas:

o Sockets API: The API may need to be extended such that a node may bind() only a subset of the available addresses, possibly by specifying a criteria (e.g. "only stable addresses", "only global", "only local", etc.).

The aforementioned work may be carried out in this document, or as a result of spin off documents.

10. 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.

11. 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.

This document also analyzes what properties are desirable for some popular scenarios, and provides advice regarding the configuration and usage of such addresses. Finally, it describes possible future standards-track work to allow for greater flexibility in IPv6 address usage.

12. 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.

13. References

13.1. 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,
            <http://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,
            <http://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, <http://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,
            <http://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,
            <http://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,
            <http://www.rfc-editor.org/info/rfc6724>.

[RFC6763]   Cheshire, S. and M. Krochmal, "DNS-Based Service
            Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013,
            <http://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,
            <http://www.rfc-editor.org/info/rfc7217>.

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

13.2. 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., 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-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-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.

   [RFC7707] Gont, F. and T. Chown, "Network Reconnaissance in IPv6
              Networks", RFC 7707, DOI 10.17487/RFC7707, March 2016,
              <http://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,
              <http://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

Madeleine Garcia Corbo
Servicios de Informacion del Transporte
Neptuno 358
Havana City 10400
Cuba

Email: madelen.garcia16@gmail.com

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

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