Internet Engineering Task Force (IETF)
Request for Comments: 7707
Obsoletes: 5157
Category: Informational
ISSN: 2070-1721

F. Gont
Huawei Technologies
T. Chown
Jisc

March 2016

Network Reconnaissance in IPv6 Networks

Abstract

IPv6 offers a much larger address space than that of its IPv4 counterpart. An IPv6 subnet of size /64 can (in theory) accommodate approximately 1.844 * 10^19 hosts, thus resulting in a much lower host density (#hosts/#addresses) than is typical in IPv4 networks, where a site typically has 65,000 or fewer unique addresses. As a result, it is widely assumed that it would take a tremendous effort to perform address-scanning attacks against IPv6 networks; therefore, IPv6 address-scanning attacks have been considered unfeasible. This document formally obsoletes RFC 5157, which first discussed this assumption, by providing further analysis on how traditional address-scanning techniques apply to IPv6 networks and exploring some additional techniques that can be employed for IPv6 network reconnaissance.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7707.

Table of Contents

1. Introduction

The main driver for IPv6 [RFC2460] deployment is its larger address space [CPNI-IPv6]. This larger address space not only allows for an increased number of connected devices but also introduces a number of subtle changes in several aspects of the resulting networks. One of these changes is the reduced host density (the number of hosts divided by the number of addresses) of typical IPv6 subnetworks, when compared to their IPv4 counterparts. [RFC5157] describes how this significantly lower IPv6 host density is likely to make classic network address-scanning attacks less feasible, since even by applying various heuristics, the address space to be scanned remains very large. RFC 5157 goes on to describe some alternative methods for attackers to glean active IPv6 addresses and provides some guidance for administrators and implementors, e.g., not using sequential addresses with DHCPv6.

With the benefit of more than five years of additional IPv6 deployment experience, this document formally obsoletes RFC 5157. It emphasizes that while address-scanning attacks are less feasible, they may, with appropriate heuristics, remain possible. At the time that RFC 5157 was written, observed address-scanning attacks were typically across ports on the addresses of discovered servers; since then, evidence that some classic address scanning is occurring is being witnessed. This text thus updates the analysis on the feasibility of address-scanning attacks in IPv6 networks, and it

explores a number of additional techniques that can be employed for IPv6 network reconnaissance. Practical examples and guidance are also included in the appendices.

On one hand, raising awareness about IPv6 network reconnaissance techniques may allow (in some cases) network and security administrators to prevent or detect such attempts. On the other hand, network reconnaissance is essential for the so-called "penetration tests" typically performed to assess the security of production networks. As a result, we believe the benefits of a thorough discussion of IPv6 network reconnaissance are twofold.

Section 4 analyzes the feasibility of address-scanning attacks (e.g., ping sweeps) in IPv6 networks and explores a number of possible improvements to such techniques. Appendix A describes how the aforementioned analysis can be leveraged to produce address-scanning tools (e.g., for penetration testing purposes). Finally, the rest of this document discusses a number of miscellaneous techniques that could be leveraged for IPv6 network reconnaissance.

2. Conventions

Throughout this document, we consider that bits are numbered from left to right, starting at 0, and that bytes are numbered from left to right, starting at 0.

3. Requirements for the Applicability of Network Reconnaissance Techniques

Throughout this document, a number of network reconnaissance techniques are discussed. Each of these techniques has different requirements on the side of the practitioner, with respect to whether they require local access to the target network and whether they require login access (or similar access credentials) to the system on which the technique is applied.

The following table tries to summarize the aforementioned requirements and serves as a cross index to the corresponding sections.

Technique	Local access	Login access
Remote Address Scanning (Section 4.2)	No	No
Local Address Scanning (Section 4.3)	Yes	No
DNS Advertised Hosts (Section 5.1.1)	No	No
DNS Zone Transfers (Section 5.1.2)	No	No
DNS Brute Forcing (Section 5.1.3)	No	No
DNS Reverse Mappings (Section 5.1.4)	No	No
Leveraging Local Name Resolution and Service Discovery Services (Section 5.2)	Yes	No
Public Archives (Section 5.3)	No	No
Application Participation (Section 5.4)	No	No
Inspection of the IPv6 Neighbor Cache and Routing Table (Section 5.5)	No	Yes
Inspecting System Configuration and Log Files (Section 5.6)	No	Yes
Gleaning Information from Routing Protocols (Section 5.7)	Yes	No
Gleaning Information from IP Flow Information Export (IPFIX) (Section 5.8)	No	Yes
Obtaining Network Information with traceroute6 (Section 5.9)	No	No
Gleaning Information from Network Devices Using SNMP (Section 5.10)	No	Yes
Obtaining Network Information via Traffic Snooping (Section 5.11)	Yes	No

Table 1: Requirements for the Applicability of Network Reconnaissance Techniques

4. IPv6 Address Scanning

This section discusses how traditional address-scanning techniques (e.g., "ping sweeps") apply to IPv6 networks. Section 4.1 provides an essential analysis of how address configuration is performed in IPv6, identifying patterns in IPv6 addresses that can be leveraged to reduce the IPv6 address search space when performing IPv6 address-scanning attacks. Section 4.2 discusses IPv6 address scanning of remote networks. Section 4.3 discusses IPv6 address scanning of local networks. Section 4.4 discusses existing IPv6 address-scanning tools. Section 4.5 provides advice on how to mitigate IPv6 address-scanning attacks. Finally, Appendix A discusses how the insights obtained in the following subsections can be incorporated into a fully fledged IPv6 address-scanning tool.

4.1. Address Configuration in IPv6

IPv6 incorporates two automatic address-configuration mechanisms: Stateless Address Autoconfiguration (SLAAC) [RFC4862] and Dynamic Host Configuration Protocol for IPv6 (DHCPv6) [RFC3315]. Support for SLAAC for automatic address configuration is mandatory, while support for DHCPv6 is optional -- however, most current versions of general-purpose operating systems support both. In addition to automatic address configuration, hosts, typically servers, may employ manual configuration, in which all the necessary information is manually entered by the host or network administrator into configuration files at the host.

The following subsections describe each of the possible configuration mechanisms/approaches in more detail.

4.1.1. Stateless Address Autoconfiguration (SLAAC)

The basic idea behind SLAAC is that every host joining a network will send a multicasted solicitation requesting network configuration information, and local routers will respond to the request providing the necessary information. SLAAC employs two different ICMPv6 message types: ICMPv6 Router Solicitation and ICMPv6 Router Advertisement messages. Router Solicitation messages are employed by hosts to query local routers for configuration information, while Router Advertisement messages are employed by local routers to convey the requested information.

Router Advertisement messages convey a plethora of network configuration information, including the IPv6 prefix that should be used for configuring IPv6 addresses on the local network. For each local prefix learned from a Router Advertisement message, an IPv6 address is configured by appending a locally generated Interface Identifier (IID) to the corresponding IPv6 prefix.

The following subsections describe currently deployed policies for generating the IIDs used with SLAAC.

4.1.1.1. Interface Identifiers Embedding IEEE Identifiers

The traditional SLAAC IIDs are based on the link-layer address of the corresponding network interface card. For example, in the case of Ethernet addresses, the IIDs are constructed as follows:

1. The "Universal" bit (bit 6, from left to right) of the address is set to 1.

2. The word 0xfffe is inserted between the Organizationally Unique Identifier (OUI) and the rest of the Ethernet address.

For example, the Media Access Control (MAC) address 00:1b:38:83:88:3c would lead to the IID 021b:38ff:fe83:883c.

A number of considerations should be made about these identifiers. Firstly, one 16-bit word (bytes 3-4) of the resulting address always has a fixed value (0xfffe), thus reducing the search space for the IID. Secondly, the high-order three bytes of the IID correspond to the OUI of the network interface card vendor. Since not all possible OUIs have been assigned, this further reduces the IID search space. Furthermore, of the assigned OUIs, many could be regarded as corresponding to legacy devices and thus are unlikely to be used for Internet-connected IPv6-enabled systems, yet further reducing the IID search space. Finally, in some scenarios, it could be possible to infer the OUI in use by the target network devices, yet narrowing down the possible IIDs even more.

NOTE:
For example, an organization known for being provisioned by vendor X is likely to have most of the nodes in its organizational network with OUIs corresponding to vendor X.

These considerations mean that in some scenarios, the original IID search space of 64 bits may be effectively reduced to 2^24 or n * 2^24 (where "n" is the number of different OUIs assigned to the target vendor).

Furthermore, if just one host address is detected or known within a subnet, it is not unlikely that, if systems were ordered in a batch, they may have sequential MAC addresses. Additionally, given a MAC address observed in one subnet, sequential or nearby MAC addresses may be seen in other subnets in the same site.

NOTE:
[RFC7136] notes that all bits of an IID should be treated as "opaque" bits. Furthermore, [DEFAULT-IIDS] is currently in the process of changing the default IID generation scheme to align with [RFC7217] (as described below in Section 4.1.1.5), such that IIDs are semantically opaque and do not follow any patterns. Therefore, the traditional IIDs based on link-layer addresses are expected to become less common over time.

4.1.1.2. Interface Identifiers of Virtualization Technologies

IIDs resulting from virtualization technologies can be considered a specific subcase of IIDs embedding IEEE identifiers (please see Section 4.1.1.1): they employ IEEE identifiers, but part of the IID has specific patterns. The following subsections describe IIDs of some popular virtualization technologies.

4.1.1.2.1. VirtualBox

All automatically generated MAC addresses in VirtualBox virtual machines employ the OUI 08:00:27 [VBox2011]. This means that all addresses resulting from traditional SLAAC will have an IID of the form a00:27ff:feXX:XXXX, thus effectively reducing the IID search space from 64 bits to 24 bits.

4.1.1.2.2. VMware ESX Server

The VMware ESX server (versions 1.0 to 2.5) provides yet a more interesting example. Automatically generated MAC addresses have the following pattern [vmesx2011]:

1. The OUI is set to 00:05:69.

2. The next 16 bits of the MAC address are set to the same value as the last 16 bits of the console operating system's primary IPv4 address.

3. The final 8 bits of the MAC address are set to a hash value based on the name of the virtual machine's configuration file.

This means that, assuming the console operating system's primary IPv4 address is known, the IID search space is reduced from 64 bits to 8 bits.

On the other hand, manually configured MAC addresses in the VMware ESX server employ the OUI 00:50:56, with the low-order three bytes of the MAC address being in the range 00:00:00-3F:FF:FF (to avoid conflicts with other VMware products). Therefore, even in the case of manually configured MAC addresses, the IID search space is reduced from 64 bits to 22 bits.

4.1.1.2.3. VMware vSphere

VMware vSphere [vSphere] supports these default MAC address generation algorithms:

- Generated addresses
  - Assigned by the vCenter server
  - Assigned by the ESXi host
- Manually configured addresses

By default, MAC addresses assigned by the vCenter server use the OUI 00:50:56 and have the format 00:50:56:XX:YY:ZZ, where XX is calculated as (0x80 + vCenter Server ID (in the range 0x00-0x3F)), and XX and YY are random two-digit hexadecimal numbers. Thus, the possible IID range is 00:50:56:80:00:00-00:50:56:BF:FF:FF; therefore, the search space for the resulting SLAAC addresses will be 22 bits.

MAC addresses generated by the ESXi host use the OUI 00:0C:29 and have the format 00:0C:29:XX:YY:ZZ, where XX, YY, and ZZ are the last three octets in hexadecimal format of the virtual machine Universally Unique Identifier (UUID) (based on a hash calculated with the UUID of the ESXi physical machine and the path to a configuration file). Thus, the MAC addresses will be in the range 00:0C:29:00:00:00-00:0C:29:FF:FF:FF; therefore, the search space for the resulting SLAAC addresses will be 24 bits.

Finally, manually configured MAC addresses employ the OUI 00:50:56, with the low-order three bytes being in the range 00:00:00-3F:FF:FF (to avoid conflicts with other VMware products). Therefore, the resulting MAC addresses will be in the range 00:50:56:00:00:00-00:50:56:3F:FF:FF, and the search space for the corresponding SLAAC addresses will be 22 bits.

4.1.1.3. Temporary Addresses

Privacy concerns [Gont-DEEPSEC2011] [RFC7721] regarding IIDs embedding IEEE identifiers led to the introduction of "Privacy Extensions for Stateless Address Autoconfiguration in IPv6" [RFC4941], also known as "temporary addresses" or "privacy addresses". Essentially, "temporary addresses" produce random addresses by concatenating a random identifier to the autoconfiguration IPv6 prefix advertised in a Router Advertisement message.

NOTE:
In addition to their unpredictability, these addresses are typically short-lived, such that even if an attacker were to learn of one of these addresses, they would be of use for a limited period of time. A typical implementation may keep a temporary address preferred for 24 hours, and configured but deprecated for seven days.

It is important to note that "temporary addresses" are generated in addition to the stable addresses [RFC7721] (such as the traditional SLAAC addresses based on IEEE identifiers): stable SLAAC addresses are meant to be employed for "server-like" inbound communications, while "temporary addresses" are meant to be employed for "client-like" outbound communications. This means that implementation/use of "temporary addresses" does not prevent an attacker from leveraging the predictability of stable SLAAC addresses, since "temporary addresses" are generated in addition to (rather than as a replacement of) the stable SLAAC addresses (such as those derived from IEEE identifiers).

The benefit that temporary addresses offer in this context is that they reduce the exposure of the host addresses to any third parties that may observe traffic sent from a host where temporary addresses are enabled and used by default. But, in the absence of firewall protection for the host, its stable SLAAC address remains liable to be scanned from off-site.

4.1.1.4. Constant, Semantically Opaque IIDs

In order to mitigate the security implications arising from the predictable IPv6 addresses derived from IEEE identifiers, Microsoft Windows produced an alternative scheme for generating "stable addresses" (in replacement of the ones embedding IEEE identifiers). The aforementioned scheme is believed to be an implementation of RFC 4941 [RFC4941], but without regenerating the addresses over time. The resulting IIDs are constant across system bootstraps, and also constant across networks.

Assuming no flaws in the aforementioned algorithm, this scheme would remove any patterns from the SLAAC addresses.

NOTE:
However, since the resulting IIDs are constant across networks, these addresses may still be leveraged for host-tracking purposes [RFC7217] [RFC7721].

The benefit of this scheme is thus that the host may be less readily detected by applying heuristics to an address-scanning attack, but, in the absence of concurrent use of temporary addresses, the host is liable to be tracked across visited networks.

4.1.1.5. Stable, Semantically Opaque IIDs

In response to the predictability issues discussed in Section 4.1.1.1 and the privacy issues discussed in [RFC7721], the IETF has standardized (in [RFC7217]) a method for generating IPv6 IIDs to be used with IPv6 SLAAC, such that addresses configured using this method are stable within each subnet, but the IIDs change when hosts move from one subnet to another. The aforementioned method is meant to be an alternative to generating IIDs based on IEEE identifiers, such that the benefits of stable addresses can be achieved without sacrificing the privacy of users.

Implementation of this method (in replacement of IIDs based on IEEE identifiers) eliminates any patterns from the IID, thus benefiting user privacy and reducing the ease with which addresses can be scanned.

4.1.2. Dynamic Host Configuration Protocol for IPv6 (DHCPv6)

DHCPv6 can be employed as a stateful address configuration mechanism, in which a server (the DHCPv6 server) leases IPv6 addresses to IPv6 hosts. As with the IPv4 counterpart, addresses are assigned according to a configuration-defined address range and policy, with some DHCPv6 servers assigning addresses sequentially, from a specific range. In such cases, addresses tend to be predictable.

NOTE:
For example, if the prefix 2001:db8::/64 is used for assigning addresses on the local network, the DHCPv6 server might (sequentially) assign addresses from the range 2001:db8::1 - 2001:db8::100.

In most common scenarios, this means that the IID search space will be reduced from the original 64 bits to 8 or 16 bits. [RFC5157] recommended that DHCPv6 instead issue addresses randomly from a large

pool; that advice is repeated here. [IIDS-DHCPv6] specifies an algorithm that can be employed by DHCPv6 servers to produce stable addresses that do not follow any specific pattern, thus resulting in an IID search space of 64 bits.

4.1.3. Manually Configured Addresses

In some scenarios, node addresses may be manually configured. This is typically the case for IPv6 addresses assigned to routers (since routers do not employ automatic address configuration) but also for servers (since having a stable address that does not depend on the underlying link-layer address is generally desirable).

While network administrators are mostly free to select the IID from any value in the range 1 - 2^64, for the sake of simplicity (i.e., ease of remembering), they tend to select addresses with one of the following patterns:

- low-byte addresses: in which most of the bytes of the IID are set to 0 (except for the least significant byte)
- IPv4-based addresses: in which the IID embeds the IPv4 address of the network interface (as in 2001:db8::192.0.2.1)
- service port addresses: in which the IID embeds the TCP/UDP service port of the main service running on that node (as in 2001:db8::80 or 2001:db8::25)
- wordy addresses: which encode words (as in 2001:db8::bad:cafe)

Each of these patterns is discussed in detail in the following subsections.

4.1.3.1. Low-Byte Addresses

The most common form of low-byte addresses is that in which all the bytes of the IID (except the least significant bytes) are set to zero (as in 2001:db8::1, 2001:db8::2, etc.). However, it is also common to find similar addresses in which the two lowest-order 16-bit words (from the right to left) are set to small numbers (as in 2001::db8::1:10, 2001:db8::2:10, etc.). Yet it is not uncommon to find IPv6 addresses in which the second lowest-order 16-bit word (from right to left) is set to a small value in the range 0x0000:0x00ff, while the lowest-order 16-bit word (from right to left) varies in the range 0x0000:0xffff. It should be noted that all of these address patterns are generally referred to as "low-byte

addresses", even when, strictly speaking, it is not only the lowest-order byte of the IPv6 address that varies from one address to another.

In the worst-case scenario, the search space for this pattern is 2^24 (although most systems can be found by searching 2^16 or even 2^8 addresses).

4.1.3.2. IPv4-Based Addresses

The most common form of these addresses is that in which an IPv4 address is encoded in the lowest-order 32 bits of the IPv6 address (usually as a result of the address notation of the form 2001:db8::192.0.2.1). However, it is also common for administrators to encode each of the bytes of the IPv4 address in each of the 16-bit words of the IID (as in, e.g., 2001:db8::192:0:2:1).

Therefore, the search space for addresses following this pattern is that of the corresponding IPv4 prefix (or twice the size of that search space if both forms of "IPv4-based addresses" are to be searched).

4.1.3.3. Service-Port Addresses

Addresses following this pattern include the service port (e.g., 80 for HTTP) in the lowest-order byte of the IID and have the rest of the bytes of the IID set to zero. There are a number of variants for this address pattern:

- The lowest-order 16-bit word (from right to left) may contain the service port, and the second lowest-order 16-bit word (from right to left) may be set to a number in the range 0x0000-0x00ff (as in, e.g., 2001:db8::1:80).

- The lowest-order 16-bit word (from right to left) may be set to a value in the range 0x0000-0x00ff, while the second lowest-order 16-bit word (from right to left) may contain the service port (as in, e.g., 2001:db8::80:1).

- The service port itself might be encoded in decimal or in hexadecimal notation (e.g., an address embedding the HTTP port might be 2001:db8::80 or 2001:db8::50) -- with addresses encoding the service port as a decimal number being more common.

Considering a maximum of 20 popular service ports, the search space for addresses following this pattern is, in the worst-case scenario, 10 * 2^11.

4.1.3.4. Wordy Addresses

Since the IPv6 address notation allows for a number of hexadecimal digits, it is not difficult to encode words into IPv6 addresses (as in, e.g., 2001:db8::bad:cafe).

Addresses following this pattern are likely to be explored by means of "dictionary attacks"; therefore, computing the corresponding search space is not straightforward.

4.1.4. IPv6 Addresses Corresponding to Transition/Coexistence Technologies

Some transition/coexistence technologies might be leveraged to reduce the target search space of remote address-scanning attacks, since they specify how the corresponding IPv6 address must be generated. For example, in the case of Teredo [RFC4380], the 64-bit IID is generated from the IPv4 address observed at a Teredo server along with a UDP port number.

For obvious reasons, the search space for these addresses will depend on the specific transition/coexistence technology being employed.

4.1.5. IPv6 Address Assignment in Real-World Network Scenarios

Figures 1, 2, and 3 provide a summary of the results obtained by [Gont-LACSEC2013] when measuring the address patterns employed by web servers, name servers, and mail servers, respectively. Figure 4 provides a rough summary of the results obtained by [Malone2008] for IPv6 routers. Figure 5 provides a summary of the results obtained by [Ford2013] for clients.

Address type	Percentage
IEEE-based	1.44%
Embedded-IPv4	25.41%
Embedded-Port	3.06%
ISATAP	0.00%
Low-byte	56.88%
Byte-pattern	6.97%
Randomized	6.24%

Figure 1: Measured Web Server Addresses

Address type	Percentage
IEEE-based	0.67%
Embedded-IPv4	22.11%
Embedded-Port	6.48%
ISATAP	0.00%
Low-byte	56.58%
Byte-pattern	11.07%
Randomized	3.09%

Figure 2: Measured Name Server Addresses

Address type	Percentage
IEEE-based	0.48%
Embedded-IPv4	4.02%
Embedded-Port	1.07%
ISATAP	0.00%
Low-byte	92.65%
Byte-pattern	1.20%
Randomized	0.59%

Figure 3: Measured Mail Server Addresses

Address type	Percentage
Low-byte	70.00%
IPv4-based	5.00%
SLAAC	1.00%
Wordy	<1.00%
Randomized	<1.00%
Teredo	<1.00%
Other	<1.00%

Figure 4: Measured Router Addresses

Address type	Percentage
IEEE-based	7.72%
Embedded-IPv4	14.31%
Embedded-Port	0.21%
ISATAP	1.06%
Randomized	69.73%
Low-byte	6.23%
Byte-pattern	0.74%

Figure 5: Measured Client Addresses

NOTE:
"ISATAP" stands for "Intra-Site Automatic Tunnel Addressing Protocol" [RFC5214].

It should be clear from these measurements that a very high percentage of host and router addresses follow very specific patterns.

Figure 5 shows that while around 70% of clients observed in this measurement appear to be using temporary addresses, a significant number of clients still expose IEEE-based addresses and addresses using embedded IPv4 (thus also revealing IPv4 addresses). Besides, as noted in Section 4.1.1.3, temporary addresses are employed along with stable IPv6 addresses; thus, hosts employing a temporary address may still be the subject of address-scanning attacks that target their stable address(es).

[ADDR-ANALYSIS] contains a spatial and temporal analysis of IPv6 addresses corresponding to clients and routers.

4.2. IPv6 Address Scanning of Remote Networks

Although attackers have been able to get away with "brute-force" address-scanning attacks in IPv4 networks (thanks to the lesser search space), successfully performing a brute-force address-scanning attack of an entire /64 network would be infeasible. As a result, it is expected that attackers will leverage the IPv6 address patterns discussed in Section 4.1 to reduce the IPv6 address search space.

IPv6 address scanning of remote networks should consider an additional factor not present for the IPv4 case: since the typical IPv6 subnet is a /64, scanning an entire /64 could, in theory, lead to the creation of 2^64 entries in the Neighbor Cache of the last-hop router. Unfortunately, a number of IPv6 implementations have been found to be unable to properly handle a large number of entries in the Neighbor Cache; hence, these address-scanning attacks may have the side effect of resulting in a Denial-of-Service (DoS) attack [CPNI-IPv6] [RFC6583].

[RFC7421] discusses the "default" /64 boundary for host subnets and the assumptions surrounding it. While there are reports of sites implementing IPv6 subnets of size /112 or smaller to reduce concerns about the above attack, such smaller subnets are likely to make address-scanning attacks more feasible, in addition to encountering the issues with non-/64 host subnets discussed in [RFC7421].

4.2.1. Reducing the Subnet ID Search Space

When address scanning a remote network, consideration is required to select which subnet IDs to choose. A typical site might have a /48 allocation, which would mean up to 65,000 or so IPv6 /64 subnets to be scanned.

However, in the same way the search space for the IID can be reduced, we may also be able to reduce the subnet ID search space in a number of ways, by guessing likely address plan schemes or using any complementary clues that might exist from other sources or observations. For example, there are a number of documents available online (e.g., [RFC5375]) that provide recommendations for the allocation of address space, which address various operational considerations, including Regional Internet Registry (RIR) assignment policy, ability to delegate reverse DNS zones to different servers, ability to aggregate routes efficiently, address space preservation, ability to delegate address assignment within the organization, ability to add/allocate new sites/prefixes to existing entities without updating Access Control Lists (ACLs), and ability to de-aggregate and advertise subspaces via various Autonomous System (AS) interfaces.

Address plans might include use of subnets that:

- Run from low ID upwards, e.g., 2001:db8:0::/64, 2001:db8:1::/64, etc.
- Use building numbers, in hexadecimal or decimal form.
- Use Virtual Local Area Network (VLAN) numbers.

- Use an IPv4 subnet number in a dual-stack target, e.g., a site with a /16 for IPv4 might use /24 subnets, and the IPv6 address plan may reuse the third byte of the IPv4 address as the IPv6 subnet ID.

- Use the service "color", as defined for service-based prefix coloring, or semantic prefixes. For example, a site using a specific coloring for a specific service such as Voice over IP (VoIP) may reduce the subnet ID search space for those devices.

The net effect is that the address space of an organization may be highly structured, and allocations of individual elements within this structure may be predictable once other elements are known.

In general, any subnet ID address plan may convey information, or be based on known information, which may in turn be of advantage to an attacker.

4.3. IPv6 Address Scanning of Local Networks

IPv6 address scanning in Local Area Networks (LANs) could be considered, to some extent, a completely different problem than that of scanning a remote IPv6 network. The main difference is that use of link-local multicast addresses can relieve the attacker of searching for unicast addresses in a large IPv6 address space.

NOTE:
While a number of other network reconnaissance vectors (such as network snooping, leveraging Neighbor Discovery traffic, etc.) are available when scanning a local network, this section focuses only on address-scanning attacks (a la "ping sweep").

An attacker can simply send probe packets to the all-nodes link-local multicast address (ff02::1), such that responses are elicited from all local nodes.

Since Windows systems (Vista, 7, etc.) do not respond to ICMPv6 Echo Request messages sent to multicast addresses, IPv6 address-scanning tools typically employ a number of additional probe packets to elicit responses from all the local nodes. For example, unrecognized IPv6 options of type 10xxxxxx elicit Internet Control Message Protocol version 6 (ICMPv6) Parameter Problem, code 2, error messages.

Many address-scanning tools discover only IPv6 link-local addresses (rather than, e.g., the global addresses of the target systems): since the probe packets are typically sent with the attacker’s IPv6 link-local address, the "victim" nodes send the response packets using the IPv6 link-local address of the corresponding network

interface (as specified by the IPv6 address-selection rules [RFC6724]). However, sending multiple probe packets, with each packet employing source addresses from different prefixes, typically helps to overcome this limitation.

4.4. Existing IPv6 Address-Scanning Tools

4.4.1. Remote IPv6 Network Address Scanners

IPv4 address-scanning tools have traditionally carried out their task by probing an entire address range (usually the entire address range comprised by the target subnetwork). One might argue that the reason for which they have been able to get away with such somewhat "rudimentary" techniques is that the scale or challenge of the task is so small in the IPv4 world that a "brute-force" attack is "good enough". However, the scale of the "address-scanning" task is so large in IPv6 that attackers must be very creative to be "good enough". Simply sweeping an entire /64 IPv6 subnet would just not be feasible.

Many address-scanning tools do not even support sweeping an IPv6 address range. On the other hand, the alive6 tool from [THC-IPV6] supports sweeping address ranges, thus being able to leverage some patterns found in IPv6 addresses, such as the incremental addresses resulting from some DHCPv6 setups. Finally, the scan6 tool from [IPv6-Toolkit] supports sweeping address ranges and can also leverage all the address patterns described in Section 4.1 of this document.

Clearly, a limitation of many of the currently available tools for IPv6 address scanning is that they lack an appropriately tuned "heuristics engine" that can help reduce the search space, such that the problem of IPv6 address scanning becomes tractable.

It should be noted that IPv6 network monitoring and management tools also need to build and maintain information about the hosts in their network. Such systems can no longer scan internal systems in a reasonable time to build a database of connected systems. Rather, such systems will need more efficient approaches, e.g., by polling network devices for data held about observed IP addresses, MAC addresses, physical ports used, etc. Such an approach can also enhance address accountability, by mapping IPv4 and IPv6 addresses to observed MAC addresses. This of course implies that any access control mechanisms for querying such network devices, e.g., community strings for SNMP, should be set appropriately to avoid an attacker being able to gather address information remotely.

4.4.2. Local IPv6 Network Address Scanners

There are a variety of publicly available local IPv6 network address-scanners:

- Current versions of nmap [nmap2015] implement this functionality.
- The Hacker's Choice (THC) IPv6 Attack Toolkit [THC-IPV6] includes a tool (alive6) that implements this functionality.
- SI6 Network's IPv6 Toolkit [IPv6-Toolkit] includes a tool (scan6) that implements this functionality.

4.5. Mitigations

IPv6 address-scanning attacks can be mitigated in a number of ways. A non-exhaustive list of the possible mitigations includes:

- Employing [RFC7217] (stable, semantically opaque IIDs) in replacement of addresses based on IEEE identifiers, such that any address patterns are eliminated.
- Employing Intrusion Prevention Systems (IPSs) at the perimeter.
- Enforcing IPv6 packet filtering where applicable (see, e.g., [RFC4890]).
- Employing manually configured MAC addresses if virtual machines are employed and "resistance" to address-scanning attacks is deemed desirable, such that even if the virtual machines employ IEEE-derived IIDs, they are generated from non-predictable MAC addresses.
- Avoiding use of sequential addresses when using DHCPv6. Ideally, the DHCPv6 server would allocate random addresses from a large pool (see, e.g., [IIDS-DHCPv6]).
- Using the "default" /64 size IPv6 subnet prefixes.
- In general, avoiding being predictable in the way addresses are assigned.

It should be noted that some of the aforementioned mitigations are operational, while others depend on the availability of specific protocol features (such as [RFC7217]) on the corresponding nodes.

Additionally, while some resistance to address-scanning attacks is generally desirable (particularly when lightweight mitigations are available), there are scenarios in which mitigation of some address-scanning vectors is unlikely to be a high priority (if at all possible). And one should always remember that security by obscurity is not a reasonable defense in itself; it may only be one (relatively small) layer in a broader security environment.

Two of the techniques discussed in this document for local address-scanning attacks are those that employ multicasted ICMPv6 Echo Requests and multicasted IPv6 packets containing unsupported options of type 10xxxxxx. These two vectors could be easily mitigated by configuring nodes to not respond to multicasted ICMPv6 Echo Requests (default on Windows systems) and by updating the IPv6 specifications (and/or possibly configuring local nodes) such that multicasted packets never elicit ICMPv6 error messages (even if they contain unsupported options of type 10xxxxxx).

NOTE:
[SMURF-AMPLIFIER] proposed such an update to the IPv6 specifications.

In any case, when it comes to local networks, there are a variety of network reconnaissance vectors. Therefore, even if address-scanning vectors were mitigated, an attacker could still rely on, e.g., protocols employed for the so-called "service discovery protocols" (see Section 5.2) or eventually rely on network snooping as a last resort for network reconnaissance. There is ongoing work in the IETF on extending mDNS, or at least DNS-based service discovery, to work across a whole site, rather than in just a single subnet, which will have associated security implications.

4.6. Conclusions

In the previous subsections, we have shown why a /64 host subnet may be more vulnerable to address-based scanning than might intuitively be thought and how an attacker might reduce the target search space when performing an address-scanning attack.

We have described a number of mitigations against address-scanning attacks, including the replacement of traditional SLAAC with stable semantically opaque IIDs (which requires support from system vendors). We have also offered some practical guidance in regard to the principle of avoiding predictability in host addressing schemes. Finally, examples of address-scanning approaches and tools are discussed in the appendices.

While most early IPv6-enabled networks remain dual stack, they are more likely to be scanned and attacked over IPv4 transport, and one may argue that the IPv6-specific considerations discussed here are not of an immediate concern. However, an early IPv6 deployment within a dual-stack network may be seen by an attacker as a potentially "easier" target if the implementation of security policies is not as strict for IPv6 (for whatever reason). As IPv6-only networks become more common, the above considerations will be of much greater importance.

5. Alternative Methods to Glean IPv6 Addresses

The following subsections describe alternative methods by which an attacker might attempt to glean IPv6 addresses for subsequent probing.

5.1. Leveraging the Domain Name System (DNS) for Network Reconnaissance

5.1.1. DNS Advertised Hosts

Any systems that are "published" in the DNS, e.g., Mail Exchange (MX) relays or web servers, will remain open to probing from the very fact that their IPv6 addresses are publicly available. It is worth noting that where the addresses used at a site follow specific patterns, publishing just one address may lead to an attack upon the other nodes.

Additionally, we note that publication of IPv6 addresses in the DNS should not discourage the elimination of IPv6 address patterns: if any address patterns are eliminated from addresses published in the DNS, an attacker may have to rely on performing dictionary-based DNS lookups in order to find all systems in a target network (which is generally less reliable and more time/traffic consuming than mapping nodes with predictable IPv6 addresses).

5.1.2. DNS Zone Transfers

A DNS zone transfer (DNS query type "AXFR") [RFC1034] [RFC1035] can readily provide information about potential attack targets. Restricting zone transfers is thus probably more important for IPv6, even if it is already good practice to restrict them in the IPv4 world.

5.1.3. DNS Brute Forcing

Attackers may employ DNS brute-forcing techniques by testing for the presence of DNS AAAA records against commonly used host names.

5.1.4. DNS Reverse Mappings

[van-Dijk] describes an interesting technique that employs DNS reverse mappings for network reconnaissance. Essentially, the attacker walks through the "ip6.arpa" zone looking up PTR records, in the hopes of learning the IPv6 addresses of hosts in a given target network (assuming that the reverse mappings have been configured, of course). What is most interesting about this technique is that it can greatly reduce the IPv6 address search space.

Basically, an attacker would walk the ip6.arpa zone corresponding to a target network (e.g., "0.8.0.0.8.b.d.0.1.0.0.2.ip6.arpa." for "2001:db8:80::/48"), issuing queries for PTR records corresponding to the domain names "0.0.8.0.0.8.b.d.0.1.0.0.2.ip6.arpa.", "1.0.8.0.0.8.b.d.0.1.0.0.2.ip6.arpa.", etc. If, say, there were PTR records for any hosts "starting" with the domain name "0.0.8.0.0.8.b.d.0.1.0.0.2.ip6.arpa." (e.g., the ip6.arpa domain name corresponding to the IPv6 address 2001:db8:80::1), the response would contain an RCODE of 0 (no error). Otherwise, the response would contain an RCODE of 4 (NXDOMAIN). As noted in [van-Dijk], this technique allows for a tremendous reduction in the "IPv6 address" search space.

NOTE:
Some name servers, incorrectly implementing the DNS protocol, reply NXDOMAIN instead of NODATA (NOERROR=0 and ANSWER=0) when encountering a domain without any resource records but that has child domains, something that is very common in ip6.arpa (these domains are called ENT for Empty Non-Terminals; see [RFC7719]). When scanning ip6.arpa, this behavior may slow down or completely prevent the exploration of ip6.arpa. Nevertheless, since such behavior is wrong (see [NXDOMAIN-DEF]), one cannot rely on it to "secure" ip6.arpa against tree walking.

[IPv6-RDNS] analyzes different approaches and considerations for ISPs in managing the ip6.arpa zone for IPv6 address space assigned to many customers, which may affect the technique described in this section.

5.2. Leveraging Local Name Resolution and Service Discovery Services

A number of protocols allow for unmanaged local name resolution and service. For example, mDNS [RFC6762] and DNS Service Discovery (DNS-SD) [RFC6763], or Link-Local Multicast Name Resolution (LLMNR) [RFC4795], are examples of such protocols.

NOTE:
Besides the Graphical User Interfaces (GUIs) included in products supporting such protocols, command-line tools such as mdns-scan [mdns-scan] and mzclient [mzclient] can help discover IPv6 hosts employing mDNS/DNS-SD.

5.3. Public Archives

Public mailing-list archives or Usenet news messages archives may prove to be a useful channel for an attacker, since hostnames and/or IPv6 addresses could be easily obtained by inspection of the (many) "Received from:" or other header lines in the archived email or Usenet news messages.

5.4. Application Participation

Peer-to-peer applications often include some centralized server that coordinates the transfer of data between peers. For example, BitTorrent [BitTorrent] builds swarms of nodes that exchange chunks of files, with a tracker passing information about peers with available chunks of data between the peers. Such applications may offer an attacker a source of peer addresses to probe.

5.5. Inspection of the IPv6 Neighbor Cache and Routing Table

Information about other systems connected to the local network might be readily available from the Neighbor Cache [RFC4861] and/or the routing table of any system connected to such network. Source Address Validation Improvement (SAVI) [RFC6620] also builds a cache of IPv6 and link-layer addresses (without actively participating in the Neighbor Discovery packet exchange) and hence is another source of similar information.

These data structures could be inspected via either "login" access or SNMP. While this requirement may limit the applicability of this technique, there are a number of scenarios in which this technique might be of use. For example, security audit tools might be provided with the necessary credentials such that the Neighbor Cache and the routing table of all systems for which the tool has "login" or SNMP access can be automatically gleaned. On the other hand, IPv6 worms [V6-WORMS] could leverage this technique for the purpose of spreading on the local network, since they will typically have access to the Neighbor Cache and routing table of an infected system.

Section 2.5.1.4 of [OPSEC-IPv6] discusses additional considerations for the inspection of the IPv6 Neighbor Cache.

5.6. Inspection of System Configuration and Log Files

Nodes are generally configured with the addresses of other important local computers, such as email servers, local file servers, web proxy servers, recursive DNS servers, etc. The /etc/hosts file in UNIX-like systems, Secure Shell (SSH) known_hosts files, or the Microsoft Windows registry are just some examples of places where interesting information about such systems might be found.

Additionally, system log files (including web server logs, etc.) may also prove to be a useful source for an attacker.

While the required credentials to access the aforementioned configuration and log files may limit the applicability of this technique, there are a number of scenarios in which this technique might be of use. For example, security audit tools might be provided with the necessary credentials such that these files can be automatically accessed. On the other hand, IPv6 worms could leverage this technique for the purpose of spreading on the local network, since they will typically have access to these files on an infected system [V6-WORMS].

5.7. Gleaning Information from Routing Protocols

Some organizational IPv6 networks employ routing protocols to dynamically maintain routing information. In such an environment, a local attacker could become a passive listener of the routing protocol, to determine other valid subnets/prefixes and some router addresses within that organization [V6-WORMS].

5.8. Gleaning Information from IP Flow Information Export (IPFIX)

IPFIX [RFC7012] can aggregate the flows by source addresses and hence may be leveraged for obtaining a list of "active" IPv6 addresses. Additional discussion of IPFIX can be found in Section 2.5.1.2 of [OPSEC-IPv6].

5.9. Obtaining Network Information with traceroute6

IPv6 traceroute [traceroute6] and similar tools (such as path6 from [IPv6-Toolkit]) can be employed to find router addresses and valid network prefixes.

5.10. Gleaning Information from Network Devices Using SNMP

SNMP can be leveraged to obtain information from a number of data structures such as the Neighbor Cache [RFC4861], the routing table, and the SAVI [RFC6620] cache of IPv6 and link-layer addresses. SNMP access should be secured, such that unauthorized access to the aforementioned information is prevented.

5.11. Obtaining Network Information via Traffic Snooping

Snooping network traffic can help in discovering active nodes in a number of ways. Firstly, each captured packet will reveal the source and destination of the packet. Secondly, the captured traffic may correspond to network protocols that transfer information such as host or router addresses, network topology information, etc.

6. Conclusions

This document explores the topic of network reconnaissance in IPv6 networks. It analyzes the feasibility of address-scanning attacks in IPv6 networks and shows that the search space for such attacks is typically much smaller than the one traditionally assumed (64 bits).

Additionally, this document explores a plethora of other network reconnaissance techniques, ranging from inspecting the IPv6 Network Cache of an attacker-controlled system to gleaning information about IPv6 addresses from public mailing-list archives or Peer-to-Peer (P2P) protocols.

We expect traditional address-scanning attacks to become more and more elaborated (i.e., less "brute force"), and other network reconnaissance techniques to be actively explored, as global deployment of IPv6 increases and, more specifically, as more IPv6-only devices are deployed.

7. Security Considerations

This document reviews methods by which addresses of hosts within IPv6 subnets can be determined. As such, it raises no new security concerns.

8. References

8.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <http://www.rfc-editor.org/info/rfc1034>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <http://www.rfc-editor.org/info/rfc1035>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <http://www.rfc-editor.org/info/rfc3315>.

[RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, DOI 10.17487/RFC4380, February 2006, <http://www.rfc-editor.org/info/rfc4380>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <http://www.rfc-editor.org/info/rfc4861>.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <http://www.rfc-editor.org/info/rfc4862>.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <http://www.rfc-editor.org/info/rfc4941>.

[RFC5214] Templin, F., Gleeson, T., and D. Thaler, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)", RFC 5214, DOI 10.17487/RFC5214, March 2008, <http://www.rfc-editor.org/info/rfc5214>.

[RFC6620] Nordmark, E., Bagnulo, M., and E. Levy-Abegnoli, "FCFS SAVI: First-Come, First-Served Source Address Validation Improvement for Locally Assigned IPv6 Addresses", RFC 6620, DOI 10.17487/RFC6620, May 2012, <http://www.rfc-editor.org/info/rfc6620>.

[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>.

[RFC7012] Claise, B., Ed. and B. Trammell, Ed., "Information Model for IP Flow Information Export (IPFIX)", RFC 7012, DOI 10.17487/RFC7012, September 2013, <http://www.rfc-editor.org/info/rfc7012>.

[RFC7136] Carpenter, B. and S. Jiang, "Significance of IPv6 Interface Identifiers", RFC 7136, DOI 10.17487/RFC7136, February 2014, <http://www.rfc-editor.org/info/rfc7136>.

[RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <http://www.rfc-editor.org/info/rfc7217>.

8.2. Informative References

[ADDR-ANALYSIS] Plonka, D. and A. Berger, "Temporal and Spatial Classification of Active IPv6 Addresses", ACM Internet Measurement Conference (IMC), Tokyo, Japan, Pages 509-522, DOI 10.1145/2815675.2815678, October 2015, <http://conferences2.sigcomm.org/imc/2015/papers/p509.pdf>.

[BitTorrent] Wikipedia, "BitTorrent", November 2015, <https://en.wikipedia.org/w/index.php?title=BitTorrent&oldid=690381343>.

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

[DEFAULT-IIDS]
Gont, F., Cooper, A., Thaler, D., and W. Liu, "Recommendation on Stable IPv6 Interface Identifiers", Work in Progress, draft-ietf-6man-default-iids-10, February 2016.

[Ford2013] Ford, M., "IPv6 Address Analysis - Privacy In, Transition Out", May 2013, <http://www.internetsociety.org/blog/2013/05/ipv6-address-analysis-privacy-transition-out>.

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

[Gont-LACSEC2013]
Gont, F., "IPv6 Network Reconnaissance: Theory & Practice", LACSEC Conference, Medellin, Colombia, May 2013, <http://www.si6networks.com/presentations/lacnic19/lacsec2013-fgont-ipv6-network-reconnaissance.pdf>.

[IIDS-DHCPv6]
Gont, F. and W. Liu, "A Method for Generating Semantically Opaque Interface Identifiers with Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", Work in Progress, draft-ietf-dhc-stable-privacy-addresses-02, April 2015.

[IPV6-EXT-HEADERS]
Gont, F., Linkova, J., Chown, T., and W. Liu, "Observations on the Dropping of Packets with IPv6 Extension Headers in the Real World", Work in Progress, draft-ietf-v6ops-ipv6-ehs-in-real-world-02, December 2015.

[IPv6-RDNS]
Howard, L., "Reverse DNS in IPv6 for Internet Service Providers", Work in Progress, draft-ietf-dnsop-isp-ip6rdns-00, October 2015.

[IPv6-Toolkit]
SI6 Networks, "SI6 Networks' IPv6 Toolkit", <http://www.si6networks.com/tools/ipv6toolkit>.

[Malone2008]
Malone, D., "Observations of IPv6 Addresses", Passive and Active Network Measurement (PAM 2008, LNCS 4979), DOI 10.1007/978-3-540-79232-1_3, April 2008, <http://www.maths.tcd.ie/~dwmalone/p/addr-pam08.pdf>.

[mdns-scan]
Poettering, L., "mdns-scan(1) Manual Page", <http://manpages.ubuntu.com/manpages/precise/man1/mdns-scan.1.html>.

[mzclient] Bockover, A., "Mono Zeroconf Project -- mzclient command-line tool", <http://www.mono-project.com/archived/monozeroconf/>.

[nmap2015] Lyon, Gordon "Fyodor", "Nmap 7.00", November 2015, <http://insecure.org>.

[NXDOMAIN-DEF]
Bortzmeyer, S. and S. Huque, "NXDOMAIN really means there is nothing underneath", Work in Progress, draft-ietf-dnsop-nxdomain-cut-00, December 2015.

[OPSEC-IPv6]
Chittimaneni, K., Kaeo, M., and E. Vyncke, "Operational Security Considerations for IPv6 Networks", Work in Progress, draft-ietf-opsec-v6-07, September 2015.

[RFC4795] Aboba, B., Thaler, D., and L. Esibov, "Link-local Multicast Name Resolution (LLMNR)", RFC 4795, DOI 10.17487/RFC4795, January 2007, <http://www.rfc-editor.org/info/rfc4795>.

[RFC4890] Davies, E. and J. Mohacsi, "Recommendations for Filtering ICMPv6 Messages in Firewalls", RFC 4890, DOI 10.17487/RFC4890, May 2007, <http://www.rfc-editor.org/info/rfc4890>.

[RFC5157] Chown, T., "IPv6 Implications for Network Scanning", RFC 5157, DOI 10.17487/RFC5157, March 2008, <http://www.rfc-editor.org/info/rfc5157>.

[RFC5375] Van de Velde, G., Popoviciu, C., Chown, T., Bonness, O., and C. Hahn, "IPv6 Unicast Address Assignment Considerations", RFC 5375, DOI 10.17487/RFC5375, December 2008, <http://www.rfc-editor.org/info/rfc5375>.

[RFC6583] Gashinsky, I., Jaeggli, J., and W. Kumari, "Operational Neighbor Discovery Problems", RFC 6583, DOI 10.17487/RFC6583, March 2012, <http://www.rfc-editor.org/info/rfc6583>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <http://www.rfc-editor.org/info/rfc6762>.

[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>.

[RFC7421] Carpenter, B., Ed., Chown, T., Gont, F., Jiang, S., Petrescu, A., and A. Yourtchenko, "Analysis of the 64-bit Boundary in IPv6 Addressing", RFC 7421, DOI 10.17487/RFC7421, January 2015, <http://www.rfc-editor.org/info/rfc7421>.

[RFC7719] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", RFC 7719, DOI 10.17487/RFC7719, December 2015, <http://www.rfc-editor.org/info/rfc7719>.

[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>.

[SMURF-AMPLIFIER] Gont, F. and W. Liu, "Security Implications of IPv6 Options of Type 10xxxxxx", Work in Progress, draft-gont-6man-ipv6-smurf-amplifier-03, March 2013.

[THC-IPV6] "THC-IPV6", <http://www.thc.org/thc-ipv6/>.

[traceroute6] FreeBSD, "FreeBSD System Manager's Manual: traceroute6(8) manual page", August 2009, <https://www.freebsd.org/cgi/man.cgi?query=traceroute6>.

[V6-WORMS] Bellovin, S., Cheswick, B., and A. Keromytis, "Worm propagation strategies in an IPv6 Internet", Vol. 31, No. 1, pp. 70-76, February 2006, <https://www.cs.columbia.edu/~smb/papers/v6worms.pdf>.

[van-Dijk] van Dijk, P., "Finding v6 hosts by efficiently mapping ip6.arpa", March 2012, <http://7bits.nl/blog/2012/03/26/finding-v6-hosts-by-efficiently-mapping-ip6-arpa>.

[VBox2011] VirtualBox, "Oracle VM VirtualBox User Manual", Version 4.1.2, August 2011, <http://www.virtualbox.org>.

[vmesx2011] VMware, "Setting a static MAC address for a virtual NIC (219)", VMware Knowledge Base, August 2011, <http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=219>.

[vSphere] VMware, "vSphere Networking", vSphere 5.5, Update 2, September 2014, <http://pubs.vmware.com/vsphere-55/topic/com.vmware.ICbase/PDF/vsphere-esxi-vcenter-server-552-networking-guide.pdf>.

Appendix A. Implementation of a Full-Fledged IPv6 Address-Scanning Tool

This section describes the implementation of a full-fledged IPv6 address-scanning tool. Appendix A.1 discusses the selection of host probes. Appendix A.2 describes the implementation of an IPv6 address scanner for local area networks. Appendix A.3 outlines the implementation of a general (i.e., non-local) IPv6 address scanner.

A.1. Host-Probing Considerations

A number of factors should be considered when selecting the probe packet types and the probing rate for an IPv6 address-scanning tool.

Firstly, some hosts (or border firewalls) might be configured to block or rate limit some specific packet types. For example, it is usual for host and router implementations to rate-limit ICMPv6 error traffic. Additionally, some firewalls might be configured to block or rate limit incoming ICMPv6 echo request packets (see, e.g., [RFC4890]).

NOTE:
As noted earlier in this document, Windows systems simply do not respond to ICMPv6 echo requests sent to multicast IPv6 addresses.

Among the possible probe types are:

- ICMPv6 Echo Request packets (meant to elicit ICMPv6 Echo Replies),
- TCP SYN segments (meant to elicit SYN/ACK or RST segments),
- TCP segments that do not contain the ACK bit set (meant to elicit RST segments),
- UDP datagrams (meant to elicit a UDP application response or an ICMPv6 Port Unreachable),
- IPv6 packets containing any suitable payload and an unrecognized extension header (meant to elicit ICMPv6 Parameter Problem error messages), or
- IPv6 packets containing any suitable payload and an unrecognized option of type 10xxxxxx (meant to elicit an ICMPv6 Parameter Problem error message).

Selecting an appropriate probe packet might help conceal the ongoing attack, but it may also be actually necessary if host or network configuration causes certain probe packets to be dropped.

Some address-scanning tools (such as scan6 of [IPv6-Toolkit]) incorporate support for IPv6 extension headers. In some cases, inserting some IPv6 extension headers in the probe packet may allow some filtering policies or monitoring devices to be circumvented. However, it may also result in the probe packets being dropped, as a result of the widespread dropping of IPv6 packets that employ IPv6 extension headers (see [IPV6-EXT-HEADERS]).

Another factor to consider is the address-probing rate. Clearly, the higher the rate, the smaller the amount of time required to perform the attack. However, the probing rate should not be too high, or else:

1. the attack might cause network congestion, thus resulting in packet loss.

2. the attack might hit rate limiting, thus resulting in packet loss.

3. the attack might reveal underlying problems in Neighbor Discovery implementations, thus leading to packet loss and possibly even Denial of Service.

Packet loss is undesirable, since it would mean that an "alive" node might remain undetected as a result of a lost probe or response. Such losses could be the result of congestion (in case the attacker is scanning a target network at a rate higher than the target network can handle) or may be the result of rate limiting (as it would be typically the case if ICMPv6 is employed for the probe packets). Finally, as discussed in [CPNI-IPv6] and [RFC6583], some IPv6 router implementations have been found to be unable to perform decent resource management when faced with Neighbor Discovery traffic involving a large number of local nodes. This essentially means that regardless of the type of probe packets, an address-scanning attack might result in a DoS of the target network, with the same (or worse) effects as that of network congestion or rate limiting.

The specific rates at which each of these issues may come into play vary from one scenario to another and depend on the type of deployed routers/firewalls, configuration parameters, etc.

A.2. Implementation of an IPv6 Local Address-Scanning Tool

scan6 [IPv6-Toolkit] is a full-fledged IPv6 local address-scanning tool, which has proven to be effective and efficient for the discovery of IPv6 hosts on a local network.

The scan6 tool operates (roughly) as follows:

1. The tool learns the local prefixes used for autoconfiguration and generates/configures one address for each local prefix (in addition to a link-local address).

2. An ICMPv6 Echo Request message destined to the all-nodes on-link multicast address (ff02::1) is sent from each of the addresses "configured" in the previous step. Because of the different source addresses, each probe packet causes the victim nodes to use different source addresses for the response packets (this allows the tool to learn virtually all the addresses in use in the local network segment).

3. The same procedure of the previous bullet is performed, but this time with ICMPv6 packets that contain an unrecognized option of type 10xxxxxx, such that ICMPv6 Parameter Problem error messages are elicited. This allows the tool to discover, e.g., Windows nodes, which otherwise do not respond to multicasted ICMPv6 Echo Request messages.

4. Each time a new "alive" address is discovered, the corresponding IID is combined with all the local prefixes, and the resulting addresses are probed (with unicasted packets). This can help to discover other addresses in use on the local network segment, since the same IID is typically used with all the available prefixes for the local network.

NOTE:
The aforementioned scheme can fail to discover some addresses for some implementations. For example, Mac OS X employs IPv6 addresses embedding IEEE identifiers (rather than "temporary addresses") when responding to packets destined to a link-local multicast address, sourced from an on-link prefix.

A.3. Implementation of an IPv6 Remote Address-Scanning Tool

An IPv6 remote address-scanning tool could be implemented with the following features:

- The tool can be instructed to target specific address ranges (e.g., 2001:db8::0-10:0-1000).

- The tool can be instructed to scan for SLAAC addresses of a specific vendor, such that only addresses embedding the corresponding IEEE OUIs are probed.

- The tool can be instructed to scan for SLAAC addresses that employ a specific IEEE OUI or set of OUIs corresponding to a specific vector.

- The tool can be instructed to discover virtual machines, such that a given IPv6 prefix is only scanned for the address patterns resulting from virtual machines.

- The tool can be instructed to scan for low-byte addresses.

- The tool can be instructed to scan for wordy addresses, in which case the tool selects addresses based on a local dictionary.

- The tool can be instructed to scan for IPv6 addresses embedding TCP/UDP service ports, in which case the tool selects addresses based on a list of well-known service ports.

- The tool can be specified to scan an IPv4 address range in use at the target network, such that only IPv4-based IPv6 addresses are scanned.

The scan6 tool of [IPv6-Toolkit] implements all these techniques/features. Furthermore, when given a target domain name or sample IPv6 address for a given prefix, the tool will try to infer the address pattern in use at the target network, and reduce the address search space accordingly.

Acknowledgements

The authors would like to thank Ray Hunter, who provided valuable text that was readily incorporated into Section 4.2.1 of this document.

The authors would like to thank (in alphabetical order) Ivan Arce, Alissa Cooper, Spencer Dawkins, Stephen Farrell, Wesley George, Marc Heuse, Ray Hunter, Barry Leiba, Libor Polcak, Alvaro Retana, Tomoyuki Sahara, Jan Schaumann, Arturo Servin, and Eric Vyncke for providing valuable comments on earlier draft versions of this document.

Fernando Gont would like to thank Jan Zorz of Go6 Lab <http://go6lab.si/> and Jared Mauch of NTT America for providing access to systems and networks that were employed to perform experiments and measurements that helped to improve this document. Additionally, he would like to thank SixXS <https://www.sixxs.net> for providing IPv6 connectivity.

Part of the contents of this document are based on the results of the project "Security Assessment of the Internet Protocol version 6 (IPv6)" [CPNI-IPv6], carried out by Fernando Gont on behalf of the UK Centre for the Protection of National Infrastructure (CPNI).

Fernando Gont would like to thank Daniel Bellomo (UNRC) for his continued support.

Authors' Addresses

Fernando Gont
Huawei Technologies
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires 1706
Argentina

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

Tim Chown
Jisc
Lumen House, Library Avenue
Harwell Oxford, Didcot. OX11 0SG
United Kingdom

Email: tim.chown@jisc.ac.uk