Operational Security Capabilities for IP Network Infrastructure (opsec)
Internet-Draft
Intended status: Informational
Expires: January 20, 2013

F. Gont
UK CPNI

July 19, 2012

Network Reconnaissance in IPv6 Networks
draft-gont-opsec-ipv6-host-scanning-01

Abstract

IPv6 offers a much larger address space than that of its IPv4 counterpart. The standard /64 IPv6 subnets can (in theory) accommodate approximately 1.844 * 10^19 hosts, thus resulting in a much lower host density (#hosts/#addresses) than their IPv4 counterparts. As a result, it is widely assumed that it would take a tremendous effort to perform address scanning attacks against IPv6 networks, and therefore IPv6 address scanning attacks have long been considered unfeasible. This document analyzes how traditional address scanning techniques apply to IPv6 networks, and also explores a number of techniques that can be employed for IPv6 network reconnaissance.

Table of Contents

1. Disclaimer

Prior work such as [RFC5157] and [V6-WORMS] still needs to be incorporated into this document. My logies -- the next rev will address this.

My understanding is that some alternative network reconnaissance techniques (DNS-based?) developed by Marc Heuse still need to be incorporated -- hopefully the next rev will address this, too.

2. Introduction

The main driver for IPv6 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 such changes is the reduced host density (Nr. of addresses/Nr. of hosts) of typical IPv6 subnetworks: with default IPv6 subnets of /64, each subnet comprises more than 1.844 * 10^19 addresses; however, the actual number of nodes in each subnet is likely to remain similar to that of IPv4 subnetworks (at most a few hundred nodes per subnet). This lower host-density has lead to the widely-established myth that IPv6 address-scanning attacks are unfeasible, since they would require a ridiculously long time (along with a tremendous amount of traffic) to be successfully performed.

This document analyzes the feasibility of "traditional" address-scanning attacks in IPv6 networks. Namely, it performs a thorough analysis of how IPv6 addresses are generated, and sheds some light on the real size of the search space for IPv6 address scanning attacks (e.g., "ping sweeps") thus dismantling the myth that such IPv6 address scanning attacks are unfeasible. Additionally, this document explores a number of other techniques, such as leveraging the DNS reverse mappings for IPv6 addresses, that can be employed for IPv6 network reconnaissance.

One 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 two-fold.

Section 3 analyzes the feasibility of traditional address-scanning attacks (e.g. ping sweeps) in IPv6 networks, and explores a number of possible improvements to such techniques. [van-Dijk] describes a recently-disclosed technique for leveraging DNS reverse mappings for discovering IPv6 nodes. Finally, Appendix A describes how the analysis carried out throughout this document can be leveraged to produce an address-scanning tools (e.g. for penetration testing purposes).

3. IPv6 Address scanning

This section discusses how traditional address scanning techniques (e.g. "ping sweeps") apply to IPv6 networks. Section 3.1 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 scans. Appendix A discusses how the insights obtained in the previous sub-sections can be incorporated into into a full-fledged IPv6 address scanning tool. Section 3.5 provides advice on how to mitigate IPv6 address scans.

3.1. Address configuration in IPv6

IPv6 incorporates two automatic address-configuration mechanisms: SLAAC (StateLess Address Auto-Configuration) [RFC4862] and DHCPv6 (Dynamic Host Configuration Protocol version 6) [RFC3315]. SLAAC is the mandatory mechanism for automatic address configuration, while DHCPv6 is optional - however, most current versions of general-purpose operating systems support both. In addition to automatic address configuration, hosts 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.

3.1.1. StateLess Address Auto-Configuration (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.

3.1.1.1. Interface-Identifiers embedding IEEE Identifiers

Many network technologies generate the 64-bit interface identifier 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 OUI (Organizationally Unique Identifier) and the rest of the Ethernet address

For example, the 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, as it should be obvious from the algorithm described above, two bytes (bytes 4-5) of the resulting address always have a fixed value (0xff, 0xfe), thus reducing the search space for the IID. Secondly, the first three bytes of these identifiers 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 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.

> 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).

Another interesting factor arises from the use of virtualization technologies, since they generally employ automatically-generated MAC addresses, with very specific patterns. For example, all automatically-generated MAC addresses in VirtualBox virtual machines employ the OUI 08:00:27 [VBox2011]. This means that all SLAAC-produced addresses will have an IID of the form a00:27ff:feXX:XXXX, thus effectively reducing the IID search space from 64 bits to 24 bits.

VMWare ESX server provides yet a more interesting example. Automatically-generated MAC addresses have the following pattern [vmesx2011]:

1. The OUI is set to 00:05:59

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 eight 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 VMWare ESX server employ the OUI 00:50:56, with the low-order three bytes being in the range 0x000000-0x3fffff (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.

3.1.1.2. Privacy Addresses

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

> In addition to their unpredictability, these addresses are typically short-lived, such that even if an attacker were to learn one of these addresses, they would be of use for a reduced period of time.

It is important to note that "privacy addresses" are generated in addition to traditional SLAAC addresses (i.e., based on IEEE identifiers): traditional SLAAC addresses are employed for incoming (i.e. server-like) communications, while "privacy addresses" are employed for outgoing (i.e., client-like) communications. This means that implementation/use of "privacy addresses" does not prevent an attacker from leveraging the predictability of traditional SLAAC addresses, since "privacy addresses" are generated in addition to (rather than in replacement of) the traditional SLAAC addresses derived from e.g. IEEE identifiers.

3.1.1.3. Stable and random Interface Identifiers

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 allegedly an implementation of RFC 4941 [RFC4941], but without regenerating the addresses over time. The resulting interface IDs 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.

> However, since the resulting interface IDs are constant across networks, these addresses may still be leveraged for host tracking purposes [I-D.ietf-6man-stable-privacy-addresses].

3.1.1.4. Stable Privacy-Enhanced Addresses

In response to the predictability issues discussed in Section 3.1.1.1 and the privacy issues discussed in , the IETF is currently standardizing (in [I-D.ietf-6man-stable-privacy-addresses]) a method for generating IPv6 Interface Identifiers to be used with IPv6 Stateless Address Autoconfiguration (SLAAC), such that addresses configured using this method are stable within each subnet, but the Interface Identifier changes when hosts move from one network to another. The aforementioned method is meant to be an alternative to generating Interface Identifiers 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 Interface Identifiers based on IEEE identifiers) would eliminate any patterns from the Interface ID.

3.1.2. Dynamic Host Configuration Protocol version 6 (DHCPv6)

DHCPv6 is 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 assigned addresses sequentially, from a specific range. In such cases, addresses tend to be predictable.

> 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.

3.1.3. 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.

While network administrators are mostly free to select the IID from any value in the range 1 - 264 range, 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 all bytes of the IID (except the lowest one) are set to 0.

- IPv4-based addresses: in which the IID encodes the IPv4-address of the network interface (as in 2001:db8::192.168.1.1)

- wordy addresses: which encode words (as in 2001:db8::dead:beef)

Clearly, the first two patterns reduce the search space from the original 64 bits to roughly 8 bits (assuming the IPv4 address range is known for the case of "IPv4-based" addresses). On the other hand, the search space for IPv6 wordy-addresses is probably larger and more complex, but still greatly reduced when compared to the original 64-bit search space.

3.1.4. IPv6 address assignment in real-world network scenarios

Table 1 and Table 2 provide a rough summary of the results obtained by [Malone2008] for IPv6 clients and IPv6 routers, respectively. These results are provided mainly for completeness-sake, since they are the most comprehensive address-measurement results that have so far been made publicly available.

We note, however, that evolution of IPv6 implementations, changes in the IPv6 address selection policy, etc., might limit (or even obsolete) the validity of these results.

Address type	Percentage
SLAAC	50%
IPv4-based	20%
Teredo	10%
Low-byte	8%
Privacy	6%
Wordy	<1%
Other	<1%

Table 1: Measured client addresses

Address type	Percentage
Low-byte	70%
IPv4-based	5%
SLAAC	1%
Wordy	<1%
Privacy	<1%
Teredo	<1%
Other	<1%

Table 2: Measured router addresses

It should be clear from these measurements that a very high percentage of the client addresses follow very specific patterns.

3.2. IPv6 address scanning of remote area networks

While in IPv4 networks attackers have been able to get away with "brute force" scanning attacks (thanks to the reduced search space), successfully performing a brute-force scan of an entire /64 network

would be infeasible. As a result, it is expected that attackers will leverage patterns found in IPv6 addresses to reduce the IPv6 address search space.

IPv6 address scanning of remote area networks should consider an additional factor not present for the IPv4 case: since the typical IPv6 subnet is a /64, this means that 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 large number of entries in the Neighbor Cache, and hence these address-scan attacks may have the side effect of resulting in a Denial of Service (DoS) attack [CPNI-IPv6] [I-D.ietf-v6ops-v6nd-problems].

3.3. IPv6 address scanning of local area networks

IPv6 address scanning in Local Area Networks 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.

> Obviously, a number of other network reconnaissance vectors (such as network snooping, leveraging Neighbor Discovery traffic, etc.) are available when scanning a local network. However, 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 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 [RFC3484]). However, sending multiple probe packets, with each packet employing addresses from different prefixes, typically helps to overcome this limitation.

This technique is employed by the scan6 tool of the IPv6 Toolkit package [IPv6-Toolkit].

3.4. Existing IPv6 address scanning tools

3.4.1. Remote IPv6 network scanners

IPv4 address scanning tools have traditionally carried out their task for probing an entire address range (usually the entire range of a target subnetwork). One might argue that the reason for which we have been able to get away with such somewhat "rudimentary" techniques is that the scale of the "problem" is so small in the IPv4 world, that a "brute-force" attack is "good enough". However, the scale of the "address scanning" problem 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. For instance, that is probably one of the reasons for which address scanning tools such as nmap [nmap2012] do not even support sweeping an IPv6 address range.

The nmap(1) manual page states "IPv6 addresses can only be specified by their fully qualified IPv6 address or hostname. CIDR and octet ranges aren't supported for IPv6 because they are rarely useful.

The most "advanced" IPv6 scanning technique that has been found in the wild is that reported in [Ybema2010], in which the attacker seemed to be scanning specific IPv6 addresses based on specific patterns. However, the aforementioned attempt probably still falls into the category of "rudimentary".

Clearly, a limitation of currently-available tools is that they lack of an "heuristics engine" that can help reduce the search space, such that the problem of IPv6 address scanning becomes tractable. However, we expect that this situation will change in the short term.

3.4.2. Local IPv6 network scanners

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

Current versions of nmap [nmap2012] implement this functionality

THC's IPv6 Attack Toolkit [THC-IPV6] includes a tool that implements this functionality

UK CPNI's IPv6 Toolkit [IPv6-Toolkit] includes a tool (scan6) that implements this functionality

3.5. Mitigations

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

- Employing stable privacy-enhanced addresses [I-D.ietf-6man-stable-privacy-addresses] in replacement of addresses based on IEEE identifiers, such that any address patterns are eliminated.

- Employing Intrusion Prevention Systems (IPS) at the perimeter, such that address scanning attacks can be mitigated.

- If virtual machines are employed, and "resistance" to address scanning attacks is deemed as desirable, manually-configured MAC addresses can be employed, such that even if the virtual machines employ IEEE-derived IIDs, they are generated from non-predictable MAC addresses.

It should be noted that some of the aforementioned mitigations are operational, while others depend on the availability of specific features (such as [I-D.ietf-6man-stable-privacy-addresses] 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).

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 Request (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).

[I-D.gont-6man-ipv6-smurf-amplifier] proposes such 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 are mitigated, an attacker could still rely on e.g. protocols employed for the so-called "opportunistic networking" (such as mDNS), or eventually on network snooping, for the purpose of network reconnaissance.

4. Leveraging DNS reverse mappings for network reconnaissance

4.1. Discussion

An interesting technique that employs DNS reverse mappings for network reconnaissance has been recently disclosed [van-Dijk]. 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:/32"), 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.

4.2. Mitigations

TBD.

5. Security Considerations

This document demonstrates that the widely-established myth of IPv6 address-scanning attacks being unfeasible is more based on "hope" than on careful analysis or facts. We expect address-scanning attacks to become more and more elaborated (i.e., less "brute force") as global deployment of IPv6 increases, and more specifically, as more IPv6-only devices are deployed.

Besides improvements in address-scanning techniques, a number of other techniques for IPv6 network reconnaissance remain to be explored. An example of some advances in this area is the use of DNS reverse mapping for discovering IPv6 nodes, as originally (and recently) described in [van-Dijk].

6. Acknowledgements

The author would like to thank (in alphabetical order) Marc Heuse, Ray Hunter, Libor Polcak, Jan Schaumann, and Arturo Servin, for providing valuable comments on earlier versions of this document.

This document resulted from 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 the UK CPNI for their continued support.

7. References

7.1. Normative References

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

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

[RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.

[RFC3484] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, February 2003.

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

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

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

[I-D.ietf-6man-stable-privacy-addresses]
Gont, F., "A method for Generating Stable Privacy-Enhanced Addresses with IPv6 Stateless Address Autoconfiguration (SLAAC)", draft-ietf-6man-stable-privacy-addresses-00 (work in progress), May 2012.

7.2. Informative References

[I-D.ietf-v6ops-v6nd-problems]
Gashinsky, I., Jaeggli, J., and W. Kumari, "Operational Neighbor Discovery Problems", draft-ietf-v6ops-v6nd-problems-05 (work in progress), March 2012.

[I-D.gont-6man-ipv6-smurf-amplifier]
Gont, F., "Security Implications of IPv6 options of Type 10xxxxxx", draft-gont-6man-ipv6-smurf-amplifier-00 (work in progress), December 2011.

[RFC5157] Chown, T., "IPv6 Implications for Network Scanning",

RFC 5157, March 2008.

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

[V6-WORMS]
Bellovin, S., Cheswick, B., and A. Keromytis, "Worm propagation strategies in an IPv6 Internet", ;login:, pages 70-76, February 2006, <https://www.cs.columbia.edu/~smb/papers/v6worms.pdf>.

[Malone2008]
Malone, D., "Observations of IPv6 Addresses", Passive and Active Measurement Conference (PAM 2008, LNCS 4979), April 2008, <http://www.maths.tcd.ie/~dwmalone/p/addr-pam08.pdf>.

[nmap2012]
Fyodor, "nmap - Network exploration tool and security / port scanner", 2012, <http://insecure.org>.

[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", vmware Knowledge Base, August 2011, <http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=219>.

[Ybema2010]
Ybema, I., "just seen my first IPv6 network abuse scan, is this the start for more?", Post to the NANOG mailing-list, 2010, <http://mailman.nanog.org/pipermail/nanog/2010-September/025049.html>.

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

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

[IPv6-Toolkit]
"IPv6 Toolkit",
<http://www.si6networks.com/research/tools.html>.

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

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 ongoing work on the implementation of a general (i.e., non-local) IPv6 host scanner.

A.1. Host-probing considerations

A number of factors should be made when selecting the probe 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.

> 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:

- o TCP segments meant to elicit SYN/ACK or RST segments,
- o UDP segments meant to elicit a UDP application response or an ICMPv6 Port Unreachable, an IPv6 packet containing any suitable payload and an unrecognized extension header (such that a ICMPv6 Parameter Problem error message is elicited), or,
- o an IPv6 packet containing any suitable payload and an unrecognized option of type 10xxxxxx (such that a ICMPv6 Parameter Problem error message is elicited)

Selecting an appropriate probe packet might help conceal the ongoing attack, but may also be actually necessary if host or network configuration causes certain probe packets to be dropped. In some cases, it might be desirable to insert some IPv6 extension headers before the actual payload, such that some filtering policies can be circumvented.

Another factor to consider is the host-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 the Neighbor Discovery implementation, 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 [I-D.ietf-v6ops-v6nd-problems], 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, a address scanning attack might result in a Denial of Service (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 prototype 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 auto-configuration, an 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 with each of the addresses "configured" in the previous step. Because of the different Source Addresses, each probe 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 Interface-ID 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 Interface ID is typically used with all the available prefixes for the local network.

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

A.3. Implementation of a IPv6 remote address-scanning tool

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

- The tool can be instructed to scan devices manufactured by a specific vendor, such that only addresses resulting for the corresponding OUIs are tried

- 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 (as discussed earlier in this document)

- The tool can be instructed to scan for low-byte or DHCPv6-like 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 specified an IPv4 address range in use at the target network, such that only IPv4-based IPv6 addresses are scanned.

In brute force mode, the tool can, at the very least:

- Skip addresses resulting from unassigned OUIs

o Skip addresses resulting from OUIs deemed as "legacy"

Author's Address

Fernando Gont
UK Centre for the Protection of National Infrastructure

Email: fernando@gont.com.ar
URI: http://www.cpni.gov.uk