Network Working Group
Internet-Draft
Intended status: Informational
Expires: July 11, 2012

W. Kumari
Google
I. Gashinsky
Yahoo!
J. Jaeggli
Zynga

January 8, 2012

Neighbor Discovery Enhancements for DOS mititgation
draft-gashinsky-6man-v6nd-enhance-00

Abstract

In IPv4, subnets are generally small, made just large enough to cover the actual number of machines on the subnet. In contrast, the default IPv6 subnet size is a /64, a number so large it covers trillions of addresses, the overwhelming number of which will be unassigned. Consequently, simplistic implementations of Neighbor Discovery can be vulnerable to denial of service attacks whereby they attempt to perform address resolution for large numbers of unassigned addresses. Such denial of attacks can be launched intentionally (by an attacker), or result from legitimate operational tools that scan networks for inventory and other purposes. As a result of these vulnerabilities, new devices may not be able to "join" a network, it may be impossible to establish new IPv6 flows, and existing ipv6 transported flows may be interrupted.

This document describes possible modifications to the traditional [RFC4861] neighbor discovery protocol for improving the resilience of the neighbor discovery process as well as an alternative method for maintaining ND caches.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 11, 2012.

Copyright Notice

Table of Contents

1. Introduction

This document describes modifications to the IPv6 Neighbor Discovery protocol [RFC4861] in order to reduce exposure to vulnerabilities when a network is scanned, either by an intruder, as part of a deliberate dos attempt, or through the use of scanning tools that peform network inventory, security audits, etc. (e.g., "nmap").

1.1. Applicability

This document is primarily intended for implementors of [RFC4861].

2. The Problem

In IPv4, subnets are generally small, made just large enough to cover the actual number of machines on the subnet. For example, an IPv4 /20 contains only 4096 address. In contrast, the default IPv6 subnet size is a /64, a number so large it covers literally billions of billions of addresses, the overwhelming number of which will be unassigned. Consequently, simplistic implementations of Neighbor Discovery can be vulnerable to denial of service attacks whereby they perform address resolution for large numbers of unassigned addresses. Such denial of attacks can be launched intentionally (by an attacker), or result from legitimate operational tools that scan networks for inventory and other purposes. As a result of these vulnerabilities, new devices may not be able to "join" a network, it may be impossible to establish new IPv6 flows, and existing ipv6 transport flows may be interrupted.

Network scans attempt to find and probe devices on a network. Typically, scans are performed on a range of target addresses, or all the addresses on a particular subnet. When such probes are directed via a router, and the target addresses are on a directly attached network, the router will to attempt to perform address resolution on a large number of destinations (i.e., some fraction of the 2^64 addresses on the subnet). The process of testing for the (non)existance of neighbors can induce a denial of service condition, where the number of Neighbor Discovery requests overwhelms the implementation's capacity to process them, exhausts available memory, replaces existing in-use mappings with incomplete entries that will never be completed, etc. The result can be network disruption, where existing traffic may be impacted, and devices that join the net find that address resolutions fails.

In order to alleviate risk associated with this DOS threat, some router implementations have taken steps to rate-limit the processing rate of Neighbor Solicitations (NS). While these mitigations do

help, they do not fully address the issue and may introduce their own set of potential liabilities to the neighbor discovery process.

3. Terminology

Address Resolution Address resolution is the process through which a node determines the link-layer address of a neighbor given only its IP address. In IPv6, address resolution is performed as part of Neighbor Discovery [RFC4861], p60

Forwarding Plane That part of a router responsible for forwarding packets. In higher-end routers, the forwarding plane is typically implemented in specialized hardware optimized for performance. Forwarding steps include determining the correct outgoing interface for a packet, decrementing its Time To Live (TTL), verifying and updating the checksum, placing the correct link-layer header on the packet, and forwarding it.

Control Plane That part of the router implementation that maintains the data structures that determine where packets should be forwarded. The control plane is typically implemented as a "slower" software process running on a general purpose processor and is responsible for such functions as the routing protocols, performing management and resolving the correct link-layer address for adjacent neighbors. The control plane "controls" the forwarding plane by programming it with the information needed for packet forwarding.

Neighbor Cache As described in [RFC4861], the data structure that holds the cache of (amongst other things) IP address to link-layer address mappings for connected nodes. The forwarding plane accesses the Neighbor Cache on every forwarded packet. Thus it is usually implemented in an ASIC .

Neighbor Discovery Process The Neighbor Discovery Process (NDP) is that part of the control plane that implements the Neighbor Discovery protocol. NDP is responsible for performing address resolution and maintaining the Neighbor Cache. When forwarding packets, the forwarding plane accesses entries within the Neighbor Cache. Whenever the forwarding plane processes a packet for which the corresponding Neighbor Cache Entry is missing or incomplete, it notifies NDP to take appropriate action (typically via a shared queue). NDP picks up requests from the shared queue and performs any necessary actions. In many implementations it is also responsible for responding to router solicitation messages, Neighbor Unreachability Detection (NUD), etc.

4. Background

Modern router architectures separate the forwarding of packets (forwarding plane) from the decisions needed to decide where the packets should go (control plane). In order to deal with the high number of packets per second the forwarding plane is generally implemented in hardware and is highly optimized for the task of forwarding packets. In contrast, the NDP control plane is mostly implemented in software processes running on a general purpose processor.

When a router needs to forward an IP packet, the forwarding plane logic performs the longest match lookup to determine where to send the packet and what outgoing interface to use. To deliver the packet to an adjacent node, It encapsulates the packet in a link-layer frame (which contains a header with the link-layer destination address). The forwarding plane logic checks the Neighbor Cache to see if it already has a suitable link-layer destination, and if not, places the request for the required information into a queue, and signals the control plane (i.e., NDP) that it needs the link-layer address resolved.

In order to protect NDP specifically and the control plane generally from being overwhelmed with these requests, appropriate steps must be taken. For example, the size and rate of the queue might be limited. NDP running in the control plane of the router dequeues requests and performs the address resolution function (by performing a neighbor solicitation and listening for a neighbor advertisement). This process is usually also responsible for other activities needed to maintain link-layer information, such as Neighbor Unreachability Detection (NUD).

An attacker sending the appropriate packets to addresses on a given subnet can cause the router to queue attempts to resolve so many addresses that it crowds out attempts to resolve "legitimate" addresses (and in many cases becomes unable to perform maintenance of existing entries in the neighbor cache, and unable to answer Neighbor Solicitiation). This condition can result the inability to resolve new neighbors and loss of reachability to neighbors with existing ND-Cache entries. During testing it was concluded that 4 simultaneous nmap sessions from a low-end computer was sufficient to make a router's neighbor discovery process unhappy and therefore forwarding unusable.

This behavior has been observed across multiple platforms and implementations.

5. Neighbor Discovery Overview

When a packet arrives at (or is generated by) a router for a destination on an attached link, the router needs to determine the correct link-layer address to send the packet to. The router checks the Neighbor Cache for an existing Neighbor Cache Entry for the neighbor, and if none exists, invokes the address resolution portions of the IPv6 Neighbor Discovery [RFC4861] protocol to determine the link-layer address.

RFC4861 Section 5.2 (Conceptual Sending Algorithm) outlines how this process works. A very high level summary is that the device creates a new Neighbor Cache Entry for the neighbor, sets the state to INCOMPLETE, queues the packet and initiates the actual address resolution process. The device then sends out one or more Neighbor Solicitiations, and when it receives a correpsonding Neighbor Advertisement, completes the Neighbor Cache Entry and sends the queued packet.

6. Recommendations for Implementors.

The section provides some recommendations to implementors of IPv4 Neighbor Discovery.

At a high-level, implementors should program defensively. That is, they should assume that intruders will attempt to exploit implementation weaknesses, and should ensure that implementations are robust to various attacks. In the case of Neighbor Discovery, the following general considerations apply:

Manage Resources Explicitely - Resources such as processor cycles, memory, etc. are never infinite, yet with IPv6's large subnets it is easy to cause NDP to generate large numbers of address resolution requests for non-existant destinations. Implementations need to limit resources devoted to processing Neighbor Discovery requests in a thoughtful manner.

Prioritize - Some NDP requests are more important than others. For example, when resources are limited, responding to Neighbor Solicitations for one's own address is more important than initiating address resolution requests that create new entries. Likewise, performing Neighbor Unreachability Detection, which by definition is only invoked on destinations that are actively being used, is more important than creating new entries for possibly non-existant neighbors.

6.1. Priortize NDP Activities

Not all Neighbor Discovery activies are equally important. Specifically, requests to perform large numbers of address resolutions on non-existant Neighbor Cache Entries should not come at the expense of servicing requests related to keeping existing, in-use entries properly up-to-date. Thus, implementations should divide work activities into categories having different priorities. The following gives examples of different activities and their importance in rough priority order.

1. It is critical to respond to Neighbor Solicitations for one's own address, especially when a router. Whether for address resolution or Neighbor Unreachability Detection, failure to respond to Neighbor Solicitations results in immediate problems. Failure to respond to NS requests that are part of NUD can cause neighbors to delete the NCE for that address, and will result in followup NS messages using multicast. Once an entry has been flushed, existing traffic for destinations using that entry can no longer be forwarded until address resolution completes succesfully. In other words, not responding to NS messages further increases the NDP load, and causes on-going communication to fail.

2. It is critical to revalidate one's own existing NCEs in need of refresh. As part of NUD, ND is required to frequently revalidate existing, in-use entries. Failure to do so can result in the entry being discarded. For in-use entries, discarding the entry will almost certainly result in a subsquent request to perform address resolution on the entry, but this time using multicast. As above, once the entry has been flushed, existing traffic for destinations using that entry can no longer be forwarded until address resolution completes succesfully.

3. To maintin the stability of the control plane, Neighbor Discovery activity related to traffic sourced by the router (as opposed to traffic being forwarded by the router) should be given high priority. Whenever network problems occur, debugging and making other operational changes requires being able to query and access the router. In addition, routing protocols may begin to react (negatively) to perceived connectivity problems, causing addition undesirable ripple effects.

4. Activities related to the sending and recieving of Router Advertisements also impact address resolutions. [XXX say more?]

5. Traffic to unknown addresses should be given lowest priority. Indeed, it may be useful to distinguish between "never seen" addresses and those that have been seen before, but that do not have

a corresponding NCE. Specifically, the conceptual processing algorithm in IPv6 Neighbor Discovery [RFC4861] calls for deleting NCEs under certain conditions. Rather than delete them completely, however, it might be useful to at least keep track of the fact that an entry at one time existed, in order to prioritize address resolution requests for such neighbors compared with neighbors that have never been seen before.

6.2. Queue Tuning.

On implementations in which requests to NDP are submitted via a single queue, router vendors SHOULD provide operators with means to control both the rate of link-layer address resolution requests placed into the queue and the size of the queue. This will allow operators to tune Neighbour Discovery for their specific environment. The ability to set or have per interface or subnet queue limits at a rate below that of the global queue limit might limit the damage to the neighbor discovery process to the taret network.

Setting those values must be a very careful balancing act - the lower the rate of entry into the queue, the less load there will be on the ND process, however, it also means that it will take the router longer to learn legitimate destinations. In a datacenter with 6,000 hosts attached to a single router, setting that value to be under 1000 would mean that resolving all of the addresses from an initial state (or something that invalidates the address cache, such as a STP TCN) may take over 6 seconds. Similarly, the lower the size of the queue, the higher the likelihood of an attack being able to knock out legitimate traffic (but less memory utilization on the router).

6.3. ND cache priming and refresh

With all of the above recommendations implemented, it should be possible to survive a "scan attack" with very little impact to the network, however, adding new hosts to the network (and the sending of traffic to them) may still be negatively impacted. Traffic to those new hosts would have to go through the unknown Neighbor Resolution queue, which is where the attack traffic would end up as well. A solution to this would be that any new host that joins the network would "announce" itself, and be added to the cache, therefore not requiring packets destined to it to go through the unknown NDP queue. This could be done by sending a ping packet to the all-routers multicast address, which would then trigger the router's own neighbor resolution process, which should be in a different queue then other packets.

All attempts should be made to keep these addresses in cache, since any eviction of legitimate hosts from the cache could potentially

place resolutions for them into the same queue as the attack traffic. At present, [RFC4861] states that there should be MAX_UNICAST_SOLICIT (3) attempts, RETRANS_TIMER 1 second apart, so if there is an interruption in the network or control plane processing for longer then 3 seconds during the refresh, the entry would be evicted from the ND Cache. Any network event which takes longer then 3 seconds to converge (UDLD, STP, etc may take 30+ seconds) while under an attack, would result in ND cache eviction. If an entry is evicted during a scan, connectivity could be lost for an extended period of time.

NDP refresh timers could be revised as suggested in draft-nordmark-6man-impatient-nud-00 [1] and SHOULD have a configurable value for MAX_UNICAST_SOLICIT and RETRANS_TIMER, and include capabilities for binary/exponential backoff.

A suggested algorithm, which retains backward compatiblity with [RFC4861] is: operator configurable values for MAX_UNICAST_SOLICIT, RETRANS_TIMER, and a way to set adaptive back-of multiple, simmilar to ipv4 -- call it BACKOFF_MULTIPLE), so that we could implement:

next_retrans = ($BACKOFF_MULTIPLE^$solicit_attempt_num)*$RETRANS_TIMER + jittered value.

The recommended behavior is to have 5 attempts, with timing spacing of 0 (initial request), 1 second later, 3 seconds later, then 9, and then 27, which represents:

MAX_UNICAST_SOLICIT=5

RETRANS_TIMER=1 (default)

BACKOFF_MULTIPLE=3

If BACKOFF_MULTIPLE=1 (which should be the default value), and MAX_UNICAST_SOLICIT=3, you would get the backwards-compatible RFC behavior, but operators should be able to adjust the values as necessary to insure that they are sufficiently aggressive about retaining ND entries in cache.

An Implementation following this algorithm would if the request was not answered at first due for example to a transitory condition, retry immediately, and then back off for progressively longer periods. This would allow for a reasonably fast resolution time when the transitory condition clears.

6.4. NDP Protocol Gratuitous NA

Per RFC 4861, section 7.2.5 and 7.2.6 [RFC4861] requires that unsolicited neighbor advertisements result in the receiver setting it's neighbor cache entry to STALE, kicking off the resolution of the neighbor using neighbor solicitation. If the link layer address in an unsolicited neighbor advertisement matches that of the existing ND cache entry, routers SHOULD retain the existing entry updating it's status with regards to LRU retention policy.

Hosts MAY be configured to send unsolicited Neighbor advertisement at a rate set at the discretion of the operators. The rate SHOULD be appropriate to the sizing of ND cache parameters and the host count on the subnet. An unsolicited NA rate parameter MUST NOT be enabled by default. The unsolicted rate interval as interpreted by hosts must jitter the value for the interval between transmissions. Hosts receiving a neighbor solicitation requests from a router following each of three subsequent gratuitous NA intervals MUST revert to RFC 4861 behavior.

Implementation of new behavior for unsolicited neighbor advertisement would make it possible under appropriate circumstances to greatly reduce the dependence on the neighbor solicitation process for retaining existing ND cache entries.

This may impact the detection of one-way reachability.

It is understood that this section may need to be moved into a separate document -- it is (currently) provided here for discussion purposes.

7. IANA Considerations

No IANA resources or consideration are requested in this draft.

8. Security Considerations

This document outlines mitigation options that operators can use to protect themselves from Denial of Service attacks. Implementation advice to router vendors aimed at ameliorating known problems carries the risk of previously unforeseen consequences. It is not believed that these techniques create additional security or DOS exposure

9. Acknowledgements

The authors would like to thank Ron Bonica, Troy Bonin, John Jason Brzozowski, Randy Bush, Vint Cerf, Jason Fesler Erik Kline, Jared Mauch, Chris Morrow and Suran De Silva. Special thanks to Thomas Narten for detailed review and (even more so) for providing text!

Apologies for anyone we may have missed; it was not intentional.

10. References

10.1. Normative References

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

[RFC4398] Josefsson, S., "Storing Certificates in the Domain Name System (DNS)", RFC 4398, March 2006.

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

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

[RFC6164] Kohno, M., Nitzan, B., Bush, R., Matsuzaki, Y., Colitti, L., and T. Narten, "Using 127-Bit IPv6 Prefixes on Inter-Router Links", RFC 6164, April 2011.

10.2. Informative References

[RFC4255] Schlyter, J. and W. Griffin, "Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints", RFC 4255, January 2006.

URIs

[1] <http://tools.ietf.org/html/draft-nordmark-6man-impatient-nud-00>

Appendix A. Text goes here.

TBD

Authors' Addresses

Warren Kumari
Google

Email: warren@kumari.net

Igor
Yahoo!
45 W 18th St
New York, NY
USA

Email: igor@yahoo-inc.com

Joel
Zynga
111 Evelyn
Sunnyvale, CA
USA

Email: jjaeggli@zynga.com