Internet Engineering Task Force
INTERNET-DRAFT
Expires: Apr 21, 2004

Craig Metz
Extreme Networks
Jun-ichiro itojun Hagino
Research Lab, IIJ
Oct 21, 2003

IPv4-Mapped Address API Considered Harmful
draft-cmetz-v6ops-v4mapped-api-harmful-01.txt

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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.''

To view the list Internet-Draft Shadow Directories, see http://www.ietf.org/shadow.html.

Distribution of this memo is unlimited.

The internet-draft will expire in 6 months. The date of expiration will be Apr 21, 2004.

Abstract

The IPv6 Addressing Architecture [Hinden, 2003] defines the "IPv4-mapped IPv6 address." This representation is used in the IPv6 basic API [Gilligan, 2003] to denote IPv4 addresses using AF_INET6 sockets. The use of IPv4-mapped addresses on AF_INET6 sockets degrades portability, complicates IPv6 systems, and is likely to create security problems. This document discusses each issue in turn. Finally, it proposes to resolve these problems by recommending deprecation of this mechanism.

1. Drawbacks due to IPv4 mapped address support

1.1. Degraded portability

RFC3493 section 3.7 specifies the behavior of IPv4-mapped addresses with an AF_INET6 socket. However, the description fails to specify important details that are necessary for good portability. Specifically, the specification needs to define:

(1) The interaction between multiple bind(2) attempts to the same port, with different addresses. What happens when an application does and does not call setsockopt(..., SO_REUSEPORT, ...) in order to bind(2) to the same port on AF_INET and AF_INET6? What happens when an application calls bind(2) on AF_INET socket, and an application calls bind(2) on an AF_INET6 socket with IPv4-mapped address? Note that there are many more issues here that need specification.

(2) The selection/interaction of port numbers between AF_INET and AF_INET6 sockets on bind(2) and/or connect(2) system calls. This is related to (1).

(3) The treatment of socket options (setsockopt(2) and getsockopt(2)) with IPv4-mapped addresses on AF_INET6 sockets. What happens when an application calls setsockopt(2) for IPv4 options or IPv6 options on an AF_INET6 socket that is not yet bound (and, therefore, the host does not know if IPv4 or IPv6 communication will be used)? What happens when an application calls setsockopt(2) for IPv4 options or IPv6 options on an AF_INET6 socket that is bound to IPv4-mapped addresses?

(4) The delivery of multicast packets to AF_INET and AF_INET6 sockets. What happens when an application binds to the IPv6 unspecified address (::) with a certain port -- does it receive IPv4 multicast traffic as well? What will be the relationship between IP_MULTICAST_IF and IPV6_MULTICAST_IF socket options? What happens when an application calls sendto(2) to an IPv4-mapped address for an IPv4 multicast address? How will the source address be selected?

Due to these ambiguities, developers of applications that use IPv4-mapped addresses on AF_INET6 sockets might encounter portability problems.

1.2. Increased implementation complexity

Implementation of IPv4-mapped addresses has a real and significant cost, both in the system software (e.g., network stack, kernel, and system libraries) and in the application software (ALL of which must now correctly handle IPv4-mapped addresses). The combined man-time for developers, testers, document writers, and support staff is a real and potentially tangible added cost of this particular feature. Because

applications are affected, the number of implementations for which this cost will apply has the potential to be huge.

Implementation of IPv4-mapped addresses increases the complexity of all IPv6 implementations, both in the system software and in the application software. Increased complexity is bad for software engineering reasons beyond the scope of this document. Technology market forces and Internet history have demonstrated that simpler protocols and simpler systems have a tendency to be more successful than complex alternatives.

If the community wishes to see IPv6 achieve successful deployment, it is important that the resource costs and complexity costs associated with IPv6 be as low as possible. Where opportunities exist to decrease these costs, the community should seriously consider doing so in order to nurture deployment.

1.3. Access control complexity

RFC3493 section 3.7 adds extra complexity to address-based access controls. It is likely that there will be many errors introduced by of this.

Due to RFC3493 section 3.7, AF_INET6 sockets will accept IPv4 packets. On an IPv4/v6 dual stack node, if there is no AF_INET listening socket, most users would believe that there will be no access from IPv4 peers. However, if an AF_INET6 listening socket is present, IPv4 peers will be able to access the service. This is likely to confuse users and result in configuration errors that can be exploited by malicious parties. (It is violating the security principle of least surprise)

AF_INET6 sockets will accept IPv4 packets even (and especially) if the application is not expecting them. Every application that uses AF_INET6 sockets MUST correctly handle reception of IPv4 packets. Failure of a developer to consider every relevant case might lead to a security problem. This is likely to be overlooked by developers -- especially those new to IPv6 development (as most are). This is likely to result in applications with flaws that can be exploited by malicious parties. (Again, this is violating the security principle of least surprise)

Systems that support IPv4 communications on AF_INET6 sockets using IPv4-mapped addresses have a greater potential to have security problems than they would if they did not have this feature.

2. Recommended solution (standard-wise)

o Deprecate RFC3493 section 3.7. By doing so, IPv6 implementations will be greatly simplified, both in the system software and in all IPv6 application software.

3. Alternative solution (standard-wise)

o Expand RFC3493 section 3.7 to fully define the behavior of AF_INET6 sockets using IPv4-mapped addresses.

o Change the default value for IPV6_V6ONLY socket option defined in [Gilligan, 2003] to "on." With this approach, systems must still implement the complex interactions between AF_INET and AF_INET6 socket, which can lead to security problems. Also, once a application turns the socket option off, it MUST correctly handle all cases where an IPv4-mapped address might be used or it may have security problems.

4. Implementation tips to application developers

o In EVERY application, check for IPv4-mapped addresses wherever addresses enter code paths under your control (i.e., are returned from system calls, or from library calls, or are input from the user or a file), and handle them in an appropriate manner. This approach is difficult in reality, and there is no way to determine whether it has been followed fully.

o Do not intentionally use RFC3493 section 3.7 (IPv4 traffic on AF_INET6 socket). Implement server applications by using separate AF_INET and AF_INET6 listening sockets. Explicitly set the IPV6_V6ONLY socket option to on, whenever the socket option is available on the system.

NOTE: Due to the lack of standard behavior in bind(2) semantics, this may not be possible on some systems. Some IPv6 stack does not permit bind(2) to 0.0.0.0, after bind(2) to ::. Also, there is no standard on how IPv4 traffic will be routed when both 0.0.0.0 and :: listening sockets are available on the same port.

NOTE: The deployment of IPV6_V6ONLY socket option is still low, not many systems support it yet.

o Implement programs in a protocol-independent manner using getaddrinfo(3) and getnameinfo(3), instead of hard-coding AF_INET6. RFC3493 section 3.7 leads people to port IPv4 application to IPv6 by replacing AF_INET into AF_INET6. However, by hard-coding AF_INET6 into the program, developers are failing to correct their dependencies on particular protocol families. As a consequence, any future protocol support will again require the application to be modified. Applications that hard-code AF_INET6 require IPv6-capable systems and will fail on a system that only has IPv4 support. It is critical to implement programs in a protocol-independent manner if you want to ship a single program (binary/source) that runs on IPv4-only, IPv6-only, as well as IPv4/v6 dual stack systems [Metz, 2000] .

5. Security considerations

This document discusses security issues with the use of IPv4-mapped

address. A recommended and alternate solution is provided.

6. Change History

00 -> 01

2553bis draft is now RFC3493. Refer RFC3513 instead of RFC2373.
Refer Craig's paper.

References

Hinden, 2003.
R. Hinden and S. Deering, "IP Version 6 Addressing Architecture" in RFC3513 (April 2003). ftp://ftp.isi.edu/in-notes/rfc3513.txt.

Gilligan, 2003.
R. Gilligan, S. Thomson, J. Bound, J. McCann, and W. R. Stevens, "Basic Socket Interface Extensions for IPv6" in RFC3493 (February 2003). ftp://ftp.isi.edu/in-notes/rfc3493.txt.

Metz, 2000.
Craig Metz, "Protocol Independence Using the Sockets API" in USENIX annual conference, Freenix track (June 2000). http://www.usenix.org/publications/library/proceedings/usenix2000/freenix/metzprotocol.html.

Author's address

Craig Metz
Extreme Networks
220 Spring Street, Suite 100
Herndon, VA 20170-5209, USA
Tel: +1 703 885 6721
email: cmetz@inner.net

Jun-ichiro itojun Hagino
Research Laboratory, Internet Initiative Japan Inc.
Takebashi Yasuda Bldg.,
3-13 Kanda Nishiki-cho,
Chiyoda-ku,Tokyo 101-0054, JAPAN
Tel: +81-3-5259-6350
Fax: +81-3-5259-6351
email: itojun@iijlab.net

^L