Internet Engineering Task Force
INTERNET-DRAFT
Expires: January 10, 2001

Jun-ichiro itojun Hagino
Research Lab, IIJ
July 10, 2000

Possible abuse against IPv6 transition technologies
draft-itojun-ipv6-transition-abuse-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.''

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at 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 January 10, 2001.

Abstract

The document talks about possible abuse of IPv6 transition technologies, which may lead to denial-of-service (DoS) attacks and other problems. IPv6 transition technologies, namely IPv6 over IPv4 tunnelling specifications and some others, have room for abuse by malicious parties. Detailed descriptions and possible workarounds are supplied.

1. Abuse of IPv4 compatible address

1.1. Problem

To implement automatic tunnelling described in RFC1933 [Gilligan, 1996] , IPv4 compatible addresses (like ::123.4.5.6) are used. From IPv6

stack point of view, an IPv4 compatible address is considered to be a normal unicast address. If an IPv6 packet has IPv4 compatible addresses in the header, the packet will be encapsulated automatically into an IPv4 packet, with IPv4 address taken from lowermost 4 bytes of the IPv4 compatible addresses. Since there is no good way to check if embedded

IPv4 address is sane, improper IPv4 packet can be generated as a result. Malicious party can abuse it, by injecting IPv6 packets to an IPv4/v6 dual stack node with certain IPv6 source address, to cause transmission of unexpected IPv4 packets. Consider the following scenario:

- o You have an IPv6 transport-capable DNS server, running on top of IPv4/v6 dual stack node. The node is on IPv4 subnet 10.1.1.0/24.

- o Malicious party transmits an IPv6 UDP packet to port 53 (DNS), with source address ::10.1.1.255. It does not make difference if it is encapsulated into an IPv4 packet, or is transmitted as a native IPv6 packet.

- o IPv6 transport-capable DNS server will transmit an IPv6 packet as a reply, copying the original source address into the destination address. Note that the IPv6 DNS server will treat IPv6 compatible address as normal IPv6 unicast address.

- o The reply packet will automatically be encapsulated into IPv4 packet, based on RFC1933 automatic tunnelling. As a result, IPv4 packet toward 10.1.1.255 will be transmitted. This is the subnet broadcast address for your IPv4 subnet, and will (improperly) reach every node on the IPv4 subnet.

1.2. Possible solutions

For the following sections, possible soluitions are presented in the order of preference (the author recommends to implement solutions that appear earlier). Note that some of the following are partial solution to the problem. Some of the solutions may overwrap, or be able to coexist, with other solutions. Solutions marked with (*) are already incorporated into [Gilligan, 2000] which is an updated version of RFC1933. Note that, however, solutions incorporated into [Gilligan, 2000] do not make a complete protection against malicious parties.

- o Disable automatic tunnelling support.

- o Reject IPv6 packets with IPv4 compatible address in IPv6 header fields. Note that we may need to check extension headers as well.

- o Perform ingress filter against IPv6 packet and tunnelled IPv6 packet. Ingress filter should let the packets with IPv4 compatible source address through, only if the source address embeds an IPv4 address belongs to the organization. The approach is a partial solution for avoiding possible transmission of malicious packet, from the organization to the outside. (*)

- o Whenever possible, check if the addresses on the packet meet the topology you have. For example, if the IPv4 address block for your site is 43.0.0.0/8, and you have a packet from IPv4-wise outside with

encapsulated IPv6 source matches ::43.0.0.0/104, it is likely that someone is doing something nasty. This may not be possible to make

the filter complete, so consider it as a partial solution. (*)

o Require use of IPv4 IPsec, namely authentication header [Kent, 1998] , for encapsulated packet. Even with IPv4 IPsec, reject the packet if the IPv6 compatible address in the IPv6 header does not embed the IPv4 address in the IPv4 header. We cannot blindly trust the inner IPv6 packet based on the existence of IPv4 IPsec association, since the inner IPv6 packet may be originated by other nodes and forwarded by the authenticated peer. The solution may be impractical, since it only solves very small part of the problem with too many requirements.

o Reject inbound/outgoing IPv6 packets, if it has certain IPv4 compatible address in IPv6 header fields. Note that we may need to check extension headers as well. The author recommends to check any IPv4 compatible address that is mapped from/to IPv4 address not suitable as IPv4 peer. They include 0.0.0.0/8, 127.0.0.0/8, 224.0.0.0/4, 255.255.255.255/32, and subnet broadcast addresses. Since the check can never be perfect (we cannot check for subnet broadcast address in remote site, for example) the direction is not recommend. (*)

2. Abuse of 6to4 address

6to4 [Carpenter, 2000] is another proposal for IPv6-over-IPv4 packet encapsulation, and is very similar to RFC1933 automatic tunneling mentioned in the previous section. 6to4 address embeds IPv4 address in the middle (2nd byte to 5th byte). If an IPv6 packet has a 6to4 address as destination address, it will be encapsulated into IPv4 packet with the embedded IPv4 address as IPv4 destination.

IPv6 packets with 6to4 address have the same problems as those with IPv4 compatible address. See the previous section for the details of the problems, and possible solutions.

The latest 6to4 draft [Carpenter, 2000] do incoporate some of the solutions presented in the previous section, however, they do not make a complete protection against malicious parties.

3. Abuse of IPv4 mapped address

3.1. Problems

IPv6 basic socket API [Gilligan, 1999] defines the use of IPv4 mapped address with AF_INET6 sockets. IPv4 mapped address is used to handle inbound IPv4 traffic toward AF_INET6 sockets, and outbound IPv4 traffic from AF_INET6 sockets. Inbound case has higher probability of abuse, while outbound case contributes to the abuse as well. Here we briefly describe the kernel behavior in inbound case. When we have an AF_INET6

socket bound to IPv6 unspecified address (::), IPv4 traffic, as well as IPv6 traffic, will be captured by the socket. The kernel will present

the address of the IPv4 peer to the userland program by using IPv4 mapped address. For example, if an IPv4 traffic from 10.1.1.1 is captured by an AF_INET6 socket, the userland program will think that the peer is at ::ffff:10.1.1.1. The userland program can manipulate IPv4 mapped address just like it would do against normal IPv6 unicast address.

We have three problems with the specification. First, IPv4 mapped address support complicates IPv4 access control mechanisms. For example, if you would like to reject accesses from IPv4 clients to a certain transport layer service, it is not enough to reject accesses to AF_INET socket. You will need to check AF_INET6 socket for accesses from IPv4 clients (seen as accesses from IPv4 mapped address) as well.

Secondly, malicious party may be able to use IPv6 packets with IPv4 mapped address, to bypass access control. Consider the following scenario:

o Attacker throws unencapsulated IPv6 packets, with ::ffff:127.0.0.1 as source address.

o The access control code in the server thinks that this is from localhost, and grants accesses.

Lastly, malicious party can make servers generate unexpected IPv4 traffic. This can be accomplished by sending IPv6 packet with IPv4 mapped address as a source (similar to abuse of IPv4 compatible address), or by presenting IPv4 mapped address to servers (like FTP bounce attack [Allman, 1999] from IPv6 to IPv4). The problem is slightly different from the problems with IPv4 compatible addresses and 6to4 addresses, since it does not make use of tunnels. It makes use of certain behavior of userland applications.

The confusion came from the dual use of IPv4 mapped address, for node-internal representation for remote IPv4 destination/source, and for real IPv6 destination/source.

3.2. Possible solutions

o In IPv6 addressing architecutre document [Hinden, 1998] , disallow the use of IPv4 mapped addresses on the wire. The change will conflict with SIIT [Nordmark, 2000] , which is the only protocol which tries to use IPv4 mapped address on IPv6 native packet. The dual use of IPv4 mapped address (as a host-internal representation of IPv4 destinations, and as a real IPv6 address) is the prime source of the problem.

o Reject IPv6 packets, if it has IPv4 mapped address in IPv6 header fields. Note that we may need to check extension headers such as routing headers, as well. IPv4 mapped address is internal

representation in a node, so doing this will raise no conflicts with existing protocols. We recommend to check the condition in IPv6 input

packet processing, and transport layer processing (TCP input and UDP input) to be sure.

- Reject DNS replies, or other host name database replies, which contain IPv4 mapped address. Again, IPv4 mapped address is internal represntation in a node and should never appear on external host name databases.

- Do not route inbound IPv4 traffic to AF_INET6 sockets. When an application would like to accept IPv4 traffic, it should explicitly open AF_INET sockets. You may want to run two applications instead, one for an AF_INET socket, and another for an AF_INET6 socket. Or you may want to make the functionality optional, off by default, and let the userland applications explicitly enable it. This greatly simplifies access control issues. This approach conflicts with what IPv6 basic API document says, however, it should raise no problem with properly-written IPv6 applications. It only affects server programs, ported by assuming the behavior of AF_INET6 listening socket against IPv4 traffic.

- When implementing TCP or UDP stack, explicitly record the wire packet format (IPv4 or IPv6) into connection table. It is unwise to guess the wire packet format, by existence of IPv6 mapped address in the address pair.

- We should separately fix problems like FTP bounce attack.

- Applications should always check if the connection to AF_INET6 socket is from an IPv4 node (IPv4 mapped address), or IPv6 node. It should then treat the connection as from IPv4 node (not from IPv6 node with special adderss), or reject the connection. This is, however, dangerous to assume that every application implementers are aware of the issue. The solution is not recommended (this is not a solution actually).

4. Attacks by combining different address formats

Malicious party can use different address formats simultaneously, in a single packet. For example, suppose you have implemented checks for abuse against IPv4 compatible address in automatic tunnel egress module. Bad guys may try to send a native IPv6 packet with 6to4 destination address with IPv4 compatible source address, to bypass security checks against IPv4 compatible address in tunnel decapsulation module. Your implementation will not be able to detect it, since the packet will not visit egress module for automatic tunnels.

Analyze code path with great care, and reject any packets that does not look sane.

5. Attacks using source address-based authentication

5.1. Problems

IPv6-to-IPv4 translators [Nordmark, 2000; Tsirtsis, 2000; Hagino, 2000] usually relay, or rewrite, IPv6 packet into IPv4 packet. The IPv4 source address in the IPv4 packet will not represent the ultimate source node (IPv6 node). Usually the IPv4 source address represents translator box instead. If we use the IPv4 source address for authentication at the destination IPv4 node, all traffic relayed/translated by the translator box will mistakenly be considered as authentic.

The problem applies to IPv4-to-IPv6 translators as well. The problem is similar to proxied services, like HTTP proxy.

5.2. Possible solutions

o Do not use translators, for protocols that use IP source address as authentication credental (for example, rlogin [Kantor, 1991]).

o translators must implement some sort of access control, to reject any IPv6 traffic from malicious IPv6 nodes.

o Do not use source address based authentication. IP source address should not be used as an authentication credental from the first place, since it is very easy for malicious parties to spoof IP source address.

6. Conclusions

IPv6 transition technologies have been proposed, however, some of them looks immune against abuse. The document presented possible ways of abuse, and possible solutions against them. The presented solutions should be reflected to the revision of specifications referenced.

For coming protocols, the author would like to propose a set of guilelines for IPv6 transition technologies:

o Tunnels must explicitly be configured. Manual configuration, or automatic configuration with proper authentication, should be okay.

o Do not embed IPv4 addresses into IPv6 addresses, for tunnels or other cases. It leaves room for abuse, since we cannot practically check if embedded IPv4 address is sane.

o Do not define an IPv6 address format that does not appear on the wire. It complicates access control issues.

The author hopes to see more deployment of native IPv6 networks, where

tunnelling technologies become unnecessary.

7. Security considerations

The document talks about security issues in existing IPv6 related protocol specifications. Possible solutions are provided.

References

Gilligan, 1996.
R. Gilligan and E. Nordmark, "Transition Mechanisms for IPv6 Hosts and Routers" in RFC1933 (April 1996). ftp://ftp.isi.edu/in-notes/rfc1933.txt.

Gilligan, 2000.
R. Gilligan and E. Nordmark, "Transition Mechanisms for IPv6 Hosts and Routers" in draft-ietf-ngtrans-mech-06.txt (April 2000). work in progress.

Kent, 1998.
S. Kent and R. Atkinson, "IP Authentication Header" in RFC2402 (November 1998). ftp://ftp.isi.edu/in-notes/rfc2402.txt.

Carpenter, 2000.
Brian Carpenter and Keith Moore, "Connection of IPv6 Domains via IPv4 Clouds without Explicit Tunnels" in draft-ietf-ngtrans-6to4-06.txt (June 2000). work in progress.

Gilligan, 1999.
R. Gilligan, S. Thomson, J. Bound, and W. Stevens, "Basic Socket Interface Extensions for IPv6" in RFC2553 (March 1999). ftp://ftp.isi.edu/in-notes/rfc2553.txt.

Allman, 1999.
M. Allman and S. Ostermann, "FTP Security Considerations" in RFC2577 (May 1999). ftp://ftp.isi.edu/in-notes/rfc2577.txt.

Hinden, 1998.
R. Hinden and S. Deering, "IP Version 6 Addressing Architecture" in RFC2373 (July 1998). ftp://ftp.isi.edu/in-notes/rfc2373.txt.

Nordmark, 2000.
E. Nordmark, "Stateless IP/ICMP Translator (SIIT)" in RFC2765 (February, 2000). ftp://ftp.isi.edu/in-notes/rfc2765.txt.

Tsirtsis, 2000.
G. Tsirtsis and P. Srisuresh, "Network Address Translation - Protocol Translation (NAT-PT)" in RFC2766 (February 2000). ftp://ftp.isi.edu/in-notes/rfc2766.txt.

Hagino, 2000.

Jun-ichiro Hagino and Kazu Yamamoto, "An IPv6-to-IPv4 transport relay translator" in draft-ietf-ngtrans-tcpudp-relay-01.txt (May 2000). work

in progress material.

Kantor, 1991.
B. Kantor, "BSD Rlogin" in RFC1282 (December 1991).
ftp://ftp.isi.edu/in-notes/rfc1282.txt.

Author's address

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