Implementation of Probabilistic Packet Marking for IPv6 Traceback

Harayama, Michiko; Kakehi, Naoyuki; and Takeuchi, Daisaku

Abstract - ***Security in computer networks has become an urgent problem. The IP traceback method is expected to prevent most vicious attacks, such as denial of service (DoS) attacks, with false source addresses in IP packets. The probabilistic packet marking (PPM) method has generated a great deal of interest in this regard.***

We herein propose an implementation method of PPM for IPv6 packets. A new hop-by-hop option of IPv6 has been defined as a trace option, whereby the information of the passing node is added. Investigations in an experimental network revealed little affect on routing performance in nodes. The present paper describes the header, the traceback algorithm, the proposed implementation method and experimental results obtained using the proposed method.

Index - ***IP traceback, IPv6, Probabilistic packet marking, Internet security.***

1. Introduction

As the Internet continues to expand, security in computer networks has become a most urgent problem. Denial of service (DoS) attacks [1] are badly damaging servers and network devices. Furthermore, DDoS (Distributed DoS) and DRDoS (Distributed Reflection DoS) attacks threaten large-scale destruction of Internet sites and services. Currently, no definitive measures exist to deal with such threats because these methods employ normal network services.

In order to prevent network attacks, Intrusion Detection Systems (IDS) [2] are used to monitor packets, and firewalls and routers filter out attacking packets. However, 'spoofing', i.e. the falsification of their source IP addresses, has rendered these systems ineffective.

1.1. Packet Tracing Methods

As effective measures for preventing DoS attacks, several packet tracing methods have been proposed. The link test method [3] traces back the path of a packet using the debugging input function of routers, but this method works only during the course of an attack. After the attack has ended, the link test method can no longer perform traces. In contrast, ITRACE [4,5] can trace the source address with ICMP traceback messages sent by routers to the destination node after an attack has ended.

Manuscript received Jun 17, 2005.

Michiko Harayama, Naoyuki Kakehi, and Daisaku Takeuchi are with the Faculty of Engineering, Gifu University, JAPAN.

Similarly, another proposal [9] traces a packet by keeping evidence about every packet in routers. In order to reduce the router load, the traceback messages are sent with very low probability, and packet-unique digested messages are logged instead of IP addresses.

Finally, the probabilistic packet marking method (PPM) [6] includes the node information in a packet as it is passing through a router.
This scheme can reduce the load of routers, and has been improved [7~10] and extended to DDoS [11].

1.2. Problem and Purpose

In PPM, the calculation time to reconstruct the path of packets increases rapidly as the number of attackers increases. In addition, most proposals up to now have been for IPv4, and the number of proposals for IPv6 is very small [9,12].

Therefore, we proposed an implementation method of PPM for IPv6 packets.
A new hop-by-hop option of IPv6 is defined as a 'trace option' in order to add information of the passing node. The present paper describes the header, the traceback algorithm, the implementation method and experimental results.

2. Probabilistic Packet Marking with IPv6

This section presents several definitions and details of probabilistic packet marking methods. In addition, this section describes the newly proposed method for IPv6.

Let a computer network be a directional graph, in which routers or computers are nodes of the graph and connections between two nodes are edges. During communication with the Internet, IP packets pass through a source node to the destination node in the graph. In PPM, every node puts its edge information in each packet to some probability as the packet is passing through.

The edge information consists of the *start* IP address, the *end* IP address, and the *distance*. The *start* IP address is the node from which the packet arrives, and the e*nd* IP address is this marking node. This pair of addresses is called the Edge-ID. *Distance* is incremented by nodes after marking, to be the hop number from the marking node to the destination node. When the packet reaches the destination, it contains the edge information. By gathering packets coming from a source node, the destination node constructs the path of the packets and finds its source address.

Although the sampling probability is very small, an adequate number of packets to construct the path will be obtained, because attackers send a sufficient number of packets to a target node during a DoS attack.

There have been a couple of proposals regarding the packet field to save the edge information. One is including the Identification field in the IPv4 header [6], and another is the inclusion of both the Type of Service field and the Identification field. Although, the Type of Service field is designed for Quality of Service (QoS), and the intended purpose of the Identification field is to recombine IPv4 fragments, these fields are not used frequently and the IPv4 header does not have any extra field space to save new types of data. For the same reason, the Flow Label field in the IPv6 header [12] is also a candidate. In fact, these fields are too short to save the edge information, which should therefore be subdivided. Subdivision of the edge information increases the calculation time required to reconstruct the path from packets [8].

By improving upon IPv4, IPv6 has various advanced functions and an effective header format [13~17]. A hop-by-hop option, one of the extended headers of IPv6, is newly defined for processes that are performed at routers. The hop-by-hop option can be used to assign adequate field size, and so it is unnecessary to subdivide the edge information. Therefore, we propose saving the edge dataset in a hop-by-hop option called the *trace option*. In the next section, the header format of the trace option and the packet marking algorithm are described.

3. Trace Option and Packet Marking Algorithm

3.1. Header Format of Trace Option

The header format of trace option is defined below and is shown in Figure 1.

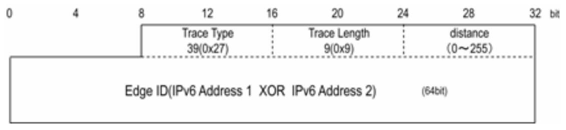

FIGURE 1: **HEADER FORMAT OF TRACE OPTION.**

Alignment Request: 8n□5
Trace Type: Type of Trace Option 39(0x27)
Trace Length: Length of Trace Option 9(0x51)

Data Field
Distance:
Distance from destination node, 0-255, 1 octet
Edge-ID:
XOR of start and end IPv6 addresses, 8 octets

The first pair of bits in Trace Type describes the procedure for routers that can not process the option.

00(0x00) neglect the option and continue processing.
01(0x40) discard the packet.
10(0x80) send an ICMP error (unknown) message.
11(0xc0) for multi-cast communication.
If not, the same as 10.

If it is possible to change the data in the option during forwarding, then the next bit should be 1. Therefore, Trace Type is 39(00100111). In the first octet of the data field, the distance between the start node and the end node, that is the current node, is stored in the next Edge-ID field, XOR of adjacent IPv6 addresses (start node and end node).

3.2. Edge Sampling Algorithm

1. When no data are stored, the router marks its own address in the Edge-ID field at p% probability and clears the Distance field by zero. Here, p=4.

2. If the distance is zero, then the router marks XOR of its own address and the data in the Edge-ID field at p% probability.

3. At step 2, the router increments the distance when an address is marked.

4. The destination node gathers several packets and inserts Edge-IDs into a directional tree in ascending order of distance to find the source address.

4. Implementation

The proposed system is implemented by modifying the KAME patch [18] under the KAME specifications.

KAME is an IPv6 project in JAPAN. The version of the KAME patch used in this system is kame-20031021-freebsd46-snap.tgz. The files in kame:kame/sys/netinet6 are modified.

A packet is sent out by the function ip6_output() at the source node. At relay nodes, the packet is received by ip6_input() and is sent out by ip6_forward(). At the destination node, the packet is received by ip6_input() and is handed over to the upper layer.

At the source node, the trace option is processed by ip6_output() and is inserted at 4% fixed probability by obtaining a pseudo-random number from microtime() in FreeBSD ver.4.6. In making the trace option, if a packet does not have a hop-by-hop option, a new hop-by-hop option header is made. If a packet already has hop-by-hop options, by obtaining a new mbuf or mbuf cluster, a trace option is inserted under the alignment requirement. Here, mbuf is the memory buffer used for handling IP headers.

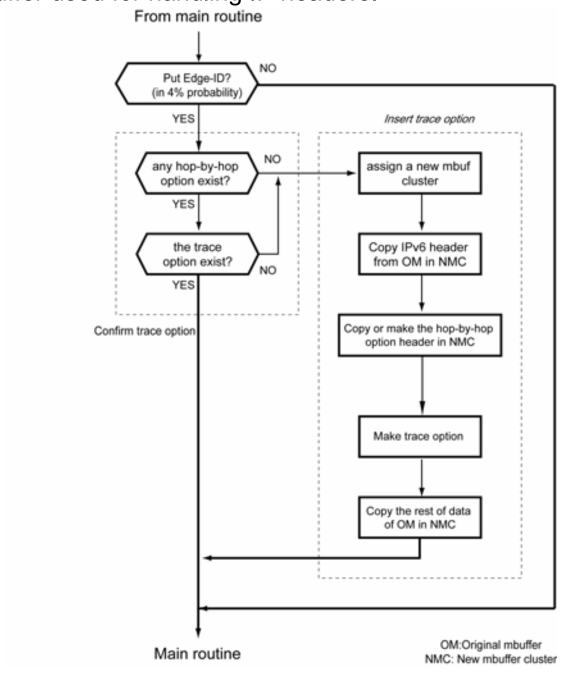
From main routine
Put Edge-ID? (in 4% probability)
NO
YES
Insert trace option
any hop-by-hop option exist?
NO
assign a new mbuf cluster
YES
the trace option exist?
NO
Copy IPv6 header from OM in NMC
YES
Confirm trace option
Copy or make the hop-by-hop option header in NMC
Make trace option
Copy the rest of data of OM in NMC
Main routine
OM:Original mbuffer
NMC: New mbuffer cluster

Figure 2: Process flow for inserting the trace option to the IPv6 header.

In contrast, at the destination node and at rely nodes, the trace option is processed by ip6_input(). The edge information of the input-side edge is entered into the trace option. The ip6_input() is used at relay nodes because the input-side edge is nearer to the source than the output-side edge.

Figure 2 shows the process flow at the relay nodes or routers. Similar to the case of the source node, the probability of insertion is determined. If a packet does not have a hop-by-hop option, a new hop-by-hop option header is made. If the packet already has a hop-by-hop option, then the packet is checked as to whether the trace option exists. If no trace option exists, then the trace option is inserted in insert_hopopts(). Here, a new mbuf cluster is assigned, to which the IPv6 header is copied. In addition, a hop-by-hop option and a trace option are inserted, after which the other data are copied.

Figure 3 shows the process for adding the edge information in the trace option. When a trace option is newly assigned, by obtaining the IPv6 address of the network interface, its upper 64 bits are stored in the Edge-ID field. The distance field is cleared by zero. If the trace option already exists, the Distance field is zero, and the probability condition is not satisfied, then the address in the Edge-ID field is replaced by the XOR of the probability condition and the probability condition of new interface. In addition, the Distance field is incremented. If the Distance field is not zero, then the field is simply incremented with no other processing.

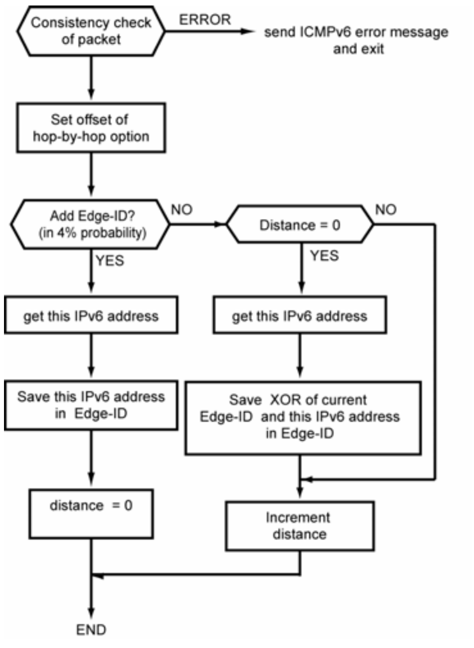
Consistency check of packet
ERROR
send ICMPv6 error message and exit
Set offset of hop-by-hop option
Add Edge-ID? (in 4% probability)
NO
Distance = 0
NO
YES
YES
get this IPv6 address
get this IPv6 address
Save this IPv6 address in Edge-ID
Save XOR of current Edge-ID and this IPv6 address in Edge-ID
distance = 0
Increment distance
END

Figure 3: Process flow for adding the edge information to the trace option.

4. EXPERIMENT AND DISCUSSION

In order to investigate the effect of this system on transmission, the response time in an experimental network is measured.

As the experimental environment, five computers were connected in series. The specifications of each computer are as follows: Pentium4 (1.5-2.6 GHz) CPU, 512 MB/MM, and FreeBSD 4.6. Transmissions among the computer were half duplex transmissions at 100 Mbps.

Both edge computers of the network work as source nodes. The other computers work as a relay node (router) and the destination node. The experimental network includes four subnetworks. Each subnetwork consists of the two adjacent computers and the cable linking them. None of the subnetworks have any common network addresses.

The modified KAME patch is installed in all of the experimental computers. The process of the trace option at each network can be skipped by manual operation in the experiment.

Table 1: Experimental Conditions (installed or not installed)

Case	Node1	Node2	Node3	Node4	Node5
No.0	No	No	No	No	No
No.1	Yes	Yes	Yes	Yes	Yes
No.2	No	Yes	Yes	Yes	Yes
No.3	Yes	Yes	No	Yes	Yes

The response time is measured for four conditions, shown in Table 1. Nodes 1-5 are the computers, and cases 1-3 are the installation conditions. The word 'yes' in the column indicates that this system is installed in the node. Table 2 shows the link number conditions.

Table 2: Experimental Conditions (link number)

link	Node1	Node2	Node3	Node4	Node5
1	Src	Dest	-	-	-
2	Src	Relay	Dest	-	-
3	Src	Relay	Relay	Dest	-
4	Src	Relay	Relay	Relay	Dest

Response time is measured by ICMPv6 echo request. One-thousand packets of five data sizes 64, 128, 256, 512, and 1024 octets, respectively, are sent three times. The experimental results are presented as the mean values of these three measurements.
Table 3 shows the response time for packets of 1024 octets.

Table 3: Response time (ms) vs. Link number

case	1	2	3	4
No.0	0.448	0.879	1.311	1.745
No.1	0.462	0.899	1.338	1.779
No.2	0.445	0.892	1.330	1.772
No.3	0.462	0.892	1.331	1.771

Figure 4 shows that the response time ratio for each case plotted against packet size. The increased response time of the system appears to be caused by increased processing time for the trace option and increased packet size. The effect is less noticeable for larger packet sizes, because of the fixed size of the trace option and the process time.

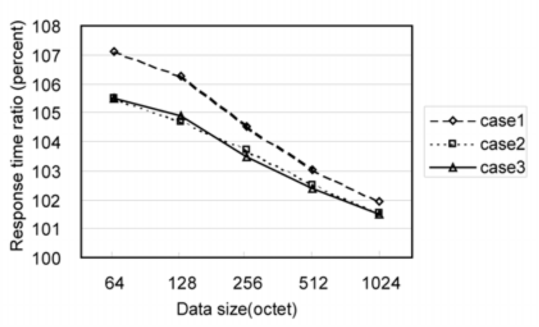

Figure 4: Response time ratio vs. Data size (all cases)

Figure 5 shows the response time ratio with respect to the link number. As the link number increases, the increase in the process time becomes smaller, and approaches a constant for three links.

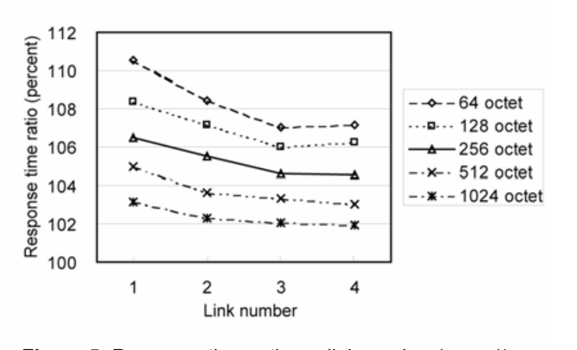

Figure 5: Response time ratio vs. link number (case 1) .

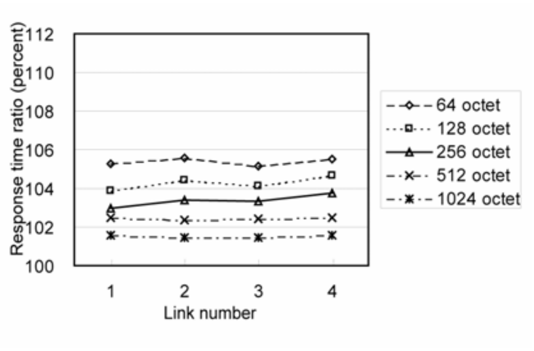

Figure 6: Response time ratio vs. link number (case 2) .

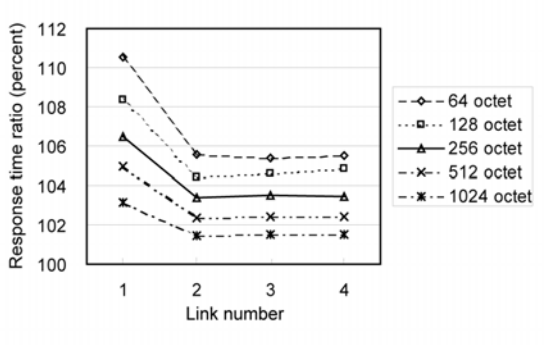

Figure 7: Response time ratio vs. link number (case 3).

Figure 6 shows the response time ratio for the case in which the source node does not assign a trace option. Comparison with Figure 5 indicates that the process of assigning a new trace option accounts for a large percentage of this increase.

Figure 7 shows the response time ratio for the case in which only the second relay node skips the trace option process. Here, the increase in process time approaches a constant with two links. These figures indicate that the response time ratio approaches a constant 2% for a packet size of 1024 octets and is 6-7% for a packet size of 64 octets.

The increase in the response time ratio is very small for large packet sizes and decreases as the link number increases for less than 2%. If packets are large and their path is long, the process time of assigning a new trace option does not seriously affect the total response time. The increased response time appears to be caused by the probability calculation, mainly with respect to the generation of random numbers. Faster random number generation will result in a greater reduction in the response time.

5. Concluding Remarks

We have proposed an implementation method of PPM for IPv6 packets. A new hop-by-hop option of IPv6 has been defined as a trace option, such that information of the passing node is added. Investigations in an experimental network reveal only a slight effect on the routing performance in nodes.

Although fixed probability is used in the algorithm described herein, the sampling probability should be determined dynamically for more effective and rapid packet trace. This will be discussed in future studies.

IPv6 traceback should be useful not only in preventing DoS/DDoS/DRDoS attacks but also for analyzing packet flow in the next-generation Internet.

REFERENCES

[1] CERT Coordination Center (CERT/CC), "CERT/CC Denial of Service", http://www.cert.org/tech_tips/denial_of_service.html

[2] The Open Source Network Intrusion Detection System, http://www.snort.org/

[3] Stone, R., "CenterTrack: An IP Overlay Network for Tracking DoS Floods", In Proceedings of USENIX Security Symposium 00,2000.

[4] Bellovin S., "The ICMP Traceback Message", http://www.research.att.com/~smb, 2000.

[5] Bellovin S., Leech M., and Taylor T., "ICMP Traceback Messages", Internet Draft (draft-ietf-itrace-03.txt),2003.

[6] Savage S., Wetherall D., Karlin A., and Anderson T., "Practical Network Support for IP Traceback", In Proceedings of SIGCOMM'00, pp.295-306, 2000.

[7] Song D. and Perrig A., "Advanced and Authenticated Marking Schemes for IP Traceback", In Proceedings of Infocom 2001, 2001.

[8] Shannon C., Moore D., and Claffy K., "Characteristics of Fragmented IP Traffic on Internet links", In Proceedings of PAM2001, 2001.

[9] Snoeren A., Partridge C., Sanchez L., Jones C., Tchakountio F., Kent S., and Strayer W., "Hash-Based IP Traceback", In Proceedings of SIGCOMM'01, 2001.

[10] Peng T., Leckie C., and Ramamohanarao K., "Adjusted Probabilistic Packet Marking for IP Traceback", In Proceedings of Networking 2002, pp.697-708, 2002.

[11] Nishino N., Harashima N., and Tokuda H., "Reflective Probabilistic Packet Marking Scheme for IP Traceback", In Proceedings of ISPJ JOURNAL Vol.44 No.8, pp.1848-1860, 2003.

[12] Oe M., "IP traceback mechanism using IPv6 Flowlabel", In Proceedings of 50th IETF itrace WG, 2001.

[13] Hagen S., "IPv6 Essentials", O'Reilly & Associates, Inc., 2002.

[14] Hinden R. and Deering S., "IP Version 6 Addressing Architecture", RFC2373, 1998.

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

[16] Borman D., Deering S., and Hinden R., "IPv6 Jumbograms", RFC2675, 1999.

[17] Partridge C. and Jackson A., "IPv6 Router Alert Option", RFC2711, 1999.

[18] KAME Project, http://www.kame.net/.

[19] Wright G.R. and Stevens W.R., "TCP/IP Illustrated, Volume2", ADDISON-WESLEY, 2000.