Regular Paper

Reflective Probabilistic Packet Marking Scheme for IP Traceback

Nobuhiko Nishio,[†1,†2] Noriyuki Harashima[†3]
and Hideyuki Tokuda[†4]

This paper describes the design and implementation of *Reflective Probabilistic Packet Marking* (RPPM) scheme, which is a traceback scheme against distributed denial-of-service (DDoS) attacks. Attacks include traffic laundered by *reflectors* which are sent false requests by attackers posing as a victim. Reflectors are among the hardest security problems on today's Internet. One promising solution to tracing the origin of attacks, the probabilistic packet marking (PPM) scheme, has proposed. However, conventional PPM cannot work against reflector attacks — *reflector problem.* Also, it encodes a mark into IP Identification field, this disables the use of ICMP — *encoding problem.* RPPM is a solution to both the reflector and encoding problem. We have extended PPM to render reflectors ineffectual by reflecting marking statistics of incoming packets at reflectors in order to trace the origin of the attacks. Furthermore, we have encoded a mark into the IP option field without reducing necessary information. Thus, RPPM can traceback beyond reflectors, ensures ICMP-compatibility, and eliminates possibility of failure in attack path reconstruction. Simulation results and our implementation based on Linux demonstrated that RPPM retains the semantics of conventional PPM on a path between an attacker and a reflector, and its performance is feasible for practice.

1. Introduction

Denial-of-service (DoS) attacks are still a major threat to the Internet and becoming more serious. Attackers are increasingly creating automated attack tools with faster deployment than ever before[1]. In DoS attacks, an attacker floods target remote machines or networks with false requests to consume their resources, thereby denying or degrading service to legitimate users. A recent observation reported over 12,000 attacks against more than 5,000 distinct targets, ranging from well-known e-commerce companies such as Amazon and Hotmail to small foreign ISPs and dial-up connections[2].

DoS attack is a serious problem and difficult to prevent, because the attackers use incorrect, or "spoofed" source addresses in the attack packets, concealing the real origin of the attacks. Unfortunately, two characteristics of the Internet allows such anonymous packets: no entity ensures that the source address is correct and no mechanism traces a path back to the origin. One approach to prevent DoS attacks is to eliminate the capability to forge the source address. One example of this approach is a filtering technique called *ingress filtering*[3] which blocks packets that arrive with illegitimate source address. The effectiveness of ingress filtering depends on universal deployment of itself. Furthermore, forged source address are legitimately used by network address translators (NATs), Mobile-IP, and various unidirectional link technologies. As the other example of this preventive approach, IPsec would be considered, if it is deployed all over the Internet. However, since IPsec is originally developed for authentication and data encryption, IPsec itself could not prevent DoS attack completely, even if it is universally deployed. Generally, for data encryption IPsec uses IKE for key exchange, and SYN packet at this time can easily lead to DoS attacks[☆]. Here, we have to clarify that we do not dare to deny any of contributions of this kind of approach for DoS attack. On the contrary, we insist that we should apply multiple techniques of different approach collaboratively against DoS attacks. In this we

†1 Graduate School of Media and Governance, Keio University
†2 "Intelligent Cooperation and Control", PRESTO, Japan Science and Technology Corporation (JST)
†3 ACCESS Co., LTD.
†4 Faculty of Environmental Information, Keio University

☆ Some may insists that routers should authenticate IPsec's AH before forwarding packets to servers by dispatching all the clients' public key to the router. It is essentially the same as ingress filter's technique and much worse in performance due the heavy to encryption processing.

mean we have to form a kind of *multiple safety net*. Therefore, we go for another approach — tracing a path back to the origin. This approach is quite different from the preventive one mentioned above. This approach is applied after attacks happened for its traceback and this also need universal deployment over the Internet for packet marking[☆].

Circumstances are worsened by further complicated distributed denial-of-service (DDoS) attacks due to use of *reflectors*[4),5)]. That is, attackers first locate a very large number of reflectors, which can be any Internet servers. Then they send the reflectors forged traffic, purportedly coming from a particular host: the victim. The reflectors will in turn generate reply traffic to the victim. The net result is that the flood at the victim arrives from numerous reflectors. Consequently, such reflector attacks also should be considered as a challenge of traceback technologies.

Although many traceback techniques have been proposed, they have significant weaknesses that limit their usability in practice. One promising solution, the probabilistic packet marking (PPM) scheme[6)] allows routers to probabilistically mark packets with partial path information during packet forwarding. The victim can then reconstruct the attack paths after receiving an estimated number of packets. However, we found the conventional PPM scheme cannot work under reflector attacks, and its encoding is incompatible with ICMP.

In this research, *Reflective Probabilistic Packet Marking* (RPPM) scheme, which is a traceback scheme against DDoS attacks including traffic laundered by reflectors, has been designed and implemented. We have extended the PPM scheme to render reflectors ineffectual by reflecting marking statistics of incoming packets at reflectors in order to trace the origin beyond the reflectors. Simulation results have shown this extension retains the semantics of conventional PPM scheme on a path between an attacker and a reflector. At the same time, we have encoded a mark into the IP option field without eliminating necessary information. This ensured that our scheme is ICMP-compatible and eliminates possibility of failure in path reconstruction.

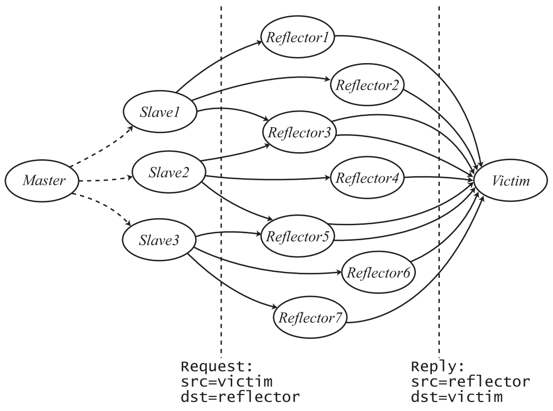

Fig. 1 Reflector attack.

The remainder of this paper is organized in the following. Reflector attack, which is one of the most difficult DoS attacks, is explained in Section 2. Section 3 describes preceding traceback techniques against DoS attacks and compares them in order to lead our design space. In Section 4, the reflector problem and encoding problem on the conventional PPM are addressed, and the design of RPPM scheme is described as a solution to these problems. Our practical implementation based on Linux is explained in Section 5, and its effectiveness and performance are shown in Section 6. Finally, we discuss our future work in Section 7 and conclude this paper in Section 8.

2. Reflector Attack

Most common DoS attacks use *IP spoofing*, where attackers forge or spoof the source address of each packet they send, thereby concealing their location and disabling an effective response. IP spoofing can also be used to make an innocent third-party "reflect" the attack. This attack, called *reflector attack*, is known as the hardest DoS attack in today's Internet[1),4)~6)]. **Figure 1** depicts reflector attacks. One host, the master attacker, manipulates compromised slaves in remote and make slaves "launder" attacks by sending a packet spoofed with a victim's address to numerous reflectors. As a result, those numerous reflectors sends responses back towards the victim. The reflectors can be any Internet server, especially DNS servers and Gnutella servers are warned.

In reflector attacks, note that individual reflector sends at a much lower rate compared to

☆ This requirement of universal deployment would be gradually regarded as an inevitable one, especially for such hosts that have global IP address and provide some services out to the Internet, considering the recent trends like various security patches are issued almost every day.

attackers flooding the victim directly, because each slave can diffuse the flooding. Suppose there are N_s slaves and N_{rf} reflectors. In general, $N_s < N_{rf}$ because the attacker can simply use reflectors by sending packets, whereas the attacker still needs to crack slaves to manipulate them. Assuming a flooding rate F coming from each slave, each reflector would generate a indirect flooding rate of $F_{rf} = \frac{N_s}{N_{rf}}F$. This indicates that slaves can reduce *indirect flooding rate* rather than *direct flooding rate*, hence they can conceal the attacks easier. In addition, the flooding rate of modern DDoS attacks is various. For example, a new DDoS method discovered by Asta[7], called *zombie pulsing attack*, directs short burst of traffic to targets. The variety of flooding rate also increases the complexity of reflector attacks.

For these reasons, the reflector attack is a serious and difficult problem. It should be dealt without loss of time, thus we address tracing the origin of DDoS attacks beyond reflectors.

3. Related Work

Over the past few years, traceback techniques against DoS attacks using IP spoofing have received considerable attention. Early traceback technique is known as testing network links at routers by input debugging. Furthermore, Burch developed the link testing concept to a novel system that automatically floods candidate network links and traces the origin of attacks[8]. However, these approaches cannot work after an attack has completed. In order to trace after an attack has completed, traceback system should store information of the attack paths in particular space. There are two approaches for such storing: *end-host approach* and *infrastructure approach*.

The end-host approach attempts to distribute the information of the attack paths in packets and collects them at end-hosts. It is explored in ITRACE proposed by Bellovin[9],[10], which lets each routers send ICMP messages to an end-host with a very low probability. After the end-host receives a sufficient number of packets, it can reconstruct the path of packets traversed. One promising scheme in the end-host approach, Savage proposed the probabilistic packet marking (PPM) scheme[6]. PPM scheme enables marking partial information of path at routers with static length field. However, the PPM scheme increases computational complexity as the number of slaves increases. Song proposed an advanced PPM scheme[11] by use of the network maps[12],[13] for reducing such computation overhead. The advantage of these end-host approaches is that once deployed, the managing cost becomes very low, because only end-hosts attempt to infer attack paths.

In contrast, the infrastructure approach attempts to store information of attack paths in the network infrastructure. Stone proposed an overlay network and achieved efficient logging[14]. Also Snoeren proposed Source Path Isolation Engine (SPIE)[15] which enhanced routers to maintain a packet digest for forwarded traffic. Since SPIE uses a large set of hash function in generating a packet digest, SPIE can traceback the origin in detail. According to SPIE, the advantage of the infrastructure approach is robustness under low volume flow (even a single packet), because it can diffuse the path information into a broad infrastructure. This characteristic is important when considering reflector attacks whose traffic volume of each attack path may be low.

However, even with these preceding traceback techniques, the reflector attack problem still remains unsolved. Since the reflector loses a mark information, the end-host approach could not deal with reflector attack. We describe the detail of this problem in Section 4.3. Also the inftrastructure approach could not regard the reflected traffic as transformed traffic, and it dismisses to trace this transformed traffic. Moreover, the infrastructure approach is expensive to manage widely distributed system rather than the end-host approach.

Consequently, we consider that the end-host approach has usability and potential ability to deal with reflector attack, however, the end-host approach has yet to be developed in practice. Therefore, we address our design space to traceback by the end-host approach.

4. Reflective Probabilistic Packet Marking Scheme

In this section, we present the design of *Reflective Probabilistic Packet Marking* (RPPM) scheme. RPPM is a traceback scheme which adopts the end-host approach and can work under DDoS attacks including traffic laundered by reflectors. We first set the goal and assumptions, then explain the two limitation of the existence PPM scheme: *reflector problem* and *encoding problem*. Next we show the solutions to these problems. Finally, we describe the detail of three algorithms, which can deal with reflector attacks: marking, reflection, and reconstruction algorithms.

4.1 The Goal

Ideally, traceback system should be able to identify a real source of packets, even if the source attempted to conceal their address by spoofing at random or using reflectors. At the same time, the traceback system should ensure backward compatibility. In traceback, we are interested in constructing *attack paths*, where the path consists of each router traversed by the packet on its journey from an attacker (either a master or a slave) to the victim. Also there are multiple source attacks, the traceback system should reconstruct an *attack graph* which is the tree of aggregated paths rooted by the victim.

Figure 2 illustrates an attack graph as viewed by a victim V. Every potential attack origin A_i is a leaf in a directed acyclic graph rooted at V, and every router R_i and reflector Rf_i are internal nodes along a path of some A and V. The attack path from A_i is the unique ordered list of routers and reflectors between A_i and V. For instance, $\{A_2, R_2, R_3, R_4, V\}$ is the attack path spoofed at random and $\{A_4, R_6, Rf_1, R_4, V\}$ is the attack path using reflectors.

To simplify the traceback problem, we groups masters and slaves in Fig. 1 as *attackers*. The detection or prevention of the path among attackers currently relies on other existing technologies [16)~18)].

4.2 Assumptions

Since the design space of traceback is large, we set the following assumptions to constrain our design space:

- an attacker can generate any packets,
- multiple attackers may conspire,

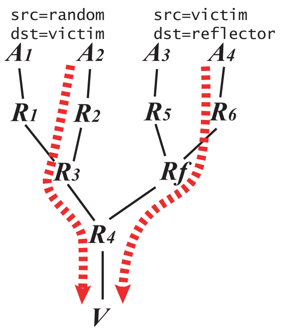

Fig. 2 An attack graph of containing two attack paths. The dotted arrows indicates attack path.

- attackers may be aware they are being traced,
- an attacker can use a reflector on each path,
- attackers may send a large number of packets during attacks,
- routers may subvert, but not often,
- the route between an attacker and a victim is fairly stable.

The first four assumptions are obvious characteristics of attacks that are restricted by the nature of the Internet. Hence, attackers can also generate a mark when they send packets, and multiple attack paths can exist with spoofing at random or using reflector.

The remaining assumptions constrain our design but need additional discussions. First, we assume that an attacker will send numerous packets to degrade the target service performance during attacks. Similar to the conventional PPM scheme, our scheme relies on this assumption, because we probabilistically let routers mark a partial information of the attack path and a victim collects packets to reconstruct an attack graph. Although the flooding rate of reflector attack is less than typical attack, the total amount of packet during attacks (that is more than an hour to keep system degrade) is still a large number. Our scheme needs only a hundred packets to reconstruct each attack path, sufficient for such attack. With regard to a single packet DoS attack, such as ping-of-death [19)], this assumption may not hold.

Second, although an attack graph may contain false positives in the presence of subverted routers, we separate with a path validation issue from traceback problem. The path validation issue is discussed more in Section 7.1. Actually, most network administrators disable the accessibility from anonymous hosts to routers, it is scarcely considered that routers will be at the disposal of attackers.

Finally, while instability of Internet routing is well-known [20)], we assume its changes extremely rare for short duration of time. By this assumption, the traceback system should shortly finish inferring an attack graph, therefore if any short-term collected packets match, it can be regard as traversed same path.

4.3 Limitation of PPM

The basic idea of the PPM scheme is that routers probabilistically write some encoding of partial path information into the packets during forwarding. This scheme reserves two static fields of the size of IP address, *start* and *end*,

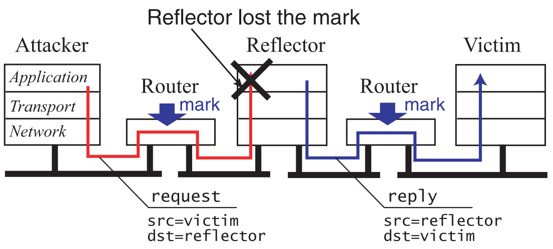

Fig. 3 A reflected attack path forwarded by the application layer on the reflector.

and a static *distance* field in each packet. Each router updates these fields as the following: each router marks the packet with a probability p. When the router decides to mark the packet, it writes its own IP address into the start field and zero into the distance field. Otherwise, if the distance field is already zero, indicating its previous router marked the packet, it writes its own IP address into the end field, representing the edge between its previous routers and itself. Finally, if the router does not mark the packet, it always increments the distance field. Thus the distance field in the packet indicates the number of routers the packet has traversed from the router which marked the packet to the victim.

However, the conventional PPM has two main problems in practice: *reflector problem* and *encoding problem*.

4.3.1 Reflector Problem

Since a reflector is not a router, received packets sent by attackers arrive at application layer on the reflector. As shown in **Fig. 3**, while the PPM scheme marks routers' addresses between the attacker and the reflector, mark information is lost if the reflector processes packets at application layer and reply to a victim as a new IP flow. Thus it impedes a traceback between attacker and reflector. The solution to this problem is described in Section 4.4.

4.3.2 Encoding Problem

Conventional PPM scheme encodes a mark into IP Identification field in the IP header. In order to use the 16-bit IP identification field, it divides each router's address with some redundancy check into k fragments. Let Ψ_d and S_d denote the set of fragment marked with a distance d and the set of router at d respectively. As a result, the computing complexity of checking combination of fragmented packets is $O(\sum_d S_d \Psi_d^k)$. However, in the case of DDoS attacks, this scheme has the following problems.

High false positives
: As d further, S_d increases. However, the number of packets marked by each router decreases, thus the combination Ψ_d cannot be identified uniquely.

High computation complexity
: As d further, S_d increases. Therefore a large number of Ψ_d combination needs to check.

Low efficiency of collecting marks
: It cannot distinguish a packet whether it is marked or not.

Song uses a large hash space to avoid a collision among numerous routers and encodes a hash ID into IP Identification field, thereby reducing computing complexity as $O(\sum_d S_d \Psi_d)$ [11]. For reducing computing overhead, it relies on the assumption that the victim has the Internet map [12] of upstream routers. Assuming the reflector attacks, such map should contain all the reflectors' upstream routers. However, its managing cost is too expensive.

One problem with changing IP Identification field is that disables IP reassembly function. Preceding schemes ignore this problem based on recent measurements which suggest only less than 0.25% of packets are fragmented [21].

Finally, the most serious problem is that changing the IP Identification field disables the use of ICMP. `ping` and `traceroute` command sends ICMP Echo and receives ICMP Echo Reply, however, it cannot recognize the reply if IP Identification field has changed. Consequently, disabling ICMP has harmful influence for such network administrating command, therefore the marking scheme should not modify IP Identification field. The solution to this problem is described in Section 4.5.

4.4 Solution to the Reflector Problem

One solution to the problem with a reflector losing a mark is to copy the mark from request to reply. For this copy, the reflector in Fig. 3 should capture the marked request packets at the network layer and preserve them. Then the reflector should also reflect the preserved mark into reply packets at network layer, hence the copying marks could be achieved.

Note the number of packets between request and reply is asymmetric. For instance on HTTP, a number of packets in GET request message is small, but those in reply message may be large. For this reason, simple marking copy cannot apply such asymmetric connection.

One possible method is use of a formula that can expect the number of packets needed to reconstruct the path bounded by $\frac{ln(d)}{p(1-p)^{d-1}}$ given by Savage[6]. Then the reflector simply collects each mark and copies them by weighting with the formula. The advantage of this method is that the number of reflected mark could be theoretic, therefore the reconstruction at a victim might be easier. However, if reflector cannot receive the sufficient number of packets, the probability distribution between before and after reflection might be significantly different, therefore the victim may fail to infer.

Alternatively, we let a reflector generate statistics for each mark, then the reflector can copy the mark in accordance with that statistics. Since the reflector should distinguish reply and request packets to generate such statistics and reflect a mark, the reflector should track the upper layer protocols not the network layer.

Smoothing time series behavior of collected packet count

Similar to the retransmission timeout calculation in TCP[22], we use *exponential smoothing*[23] to reduce irregularities in time series behavior of packet count. Let t and $Y(t)$ denote the time domain and the observation for at time series data of packet count respectively. Exponential smoothing generates an estimate $S(t)$ from the new observation $Y(t)$ and the old estimate $S(t-1)$ with a fraction α called *smoothing constant*. In practice, $S(t)$ is calculated recursively in the following fashion.

$$S(t) = \alpha Y(t) + (1-\alpha)S(t-1) \qquad (1)$$

Bowerman stated that the smoothing constant between 0.01 to 0.3 usually worked quite well in practice[24]. In our experience, the smoothing constant $\alpha = 0.3$ was the best value.

Furthermore, as we mentioned in Section 2, the flooding rate of DDoS attacks is various. Hence, the sampling period T of statistics needs to calculate for exponential smoothing. Since $S(t)$ is the positive integer for our scheme, if assumed P as the necessary precision number, then the sampling period of statistics should be chosen as $\alpha Y(t) > 10^P$. Let F denote the flooding rate per second, it leads $\alpha TFp(1-p)^{d-1} > 10^P$. Consequently, the recommended T can be represented as following.

$$T > \frac{10^P}{\alpha Fp(1-p)^{d-1}} \qquad (2)$$

If $\alpha = 0.3$, $P = 2$, $F = 1000$, $p = 0.05$, $d = 10$, the sampling period of statistics should be more than 11 seconds.

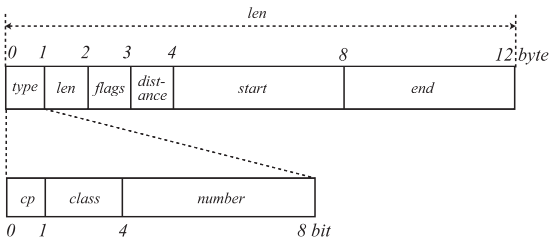

Fig. 4 Format of IP option RPPM.

The valuable T can make appropriate reflection at reflector for any rate of DDoS attack. For example, assuming short burst of traffic, such as zombie pulsing attack, statistics can moderate the irregularities of packet marking.

4.5 Solution to the Encoding Problem

To ensure the backward compatibility, we propose using IP option for packet marking. Although preceding schemes dismissed IP option approach as a poor choice due to the expense of appending additional data on flight, it is obvious that practical functionality is important over performance. Actually, we consider our approach scarcely degrades the system by the following reasons:

- most routers have become powerful enough to process IP options,
- it is extremely rare that IP option are used (e.g., `ping` with IP Record Route option[25]).

For these reasons, we designed an IP option Reflective Probabilistic Packet Marking (IP option RPPM) as shown in **Fig. 4**. The first byte *type* is divided into three internal fields: a 1-bit *cp* flag, a 2-bit *class* field, and a 5-bit *number* field. The *cp* flag indicates whether the individual option should be copied into the IP header of the fragments. To reduce the overhead, we defined 0 in order not to copy. The *class* field groups related options and we choose most general class *control* defined 0. In the *number field*, we arbitrarily defined a new number 28. The second byte is *len* field, that covers the *type*, *len*, and remaining bytes. In our scheme, it is statically 12. The third byte has not yet defined and simply used for IP header padding, thus this 8-bit fields can be defined for extensional flags. The remaining bytes of static *distance*, *start*, and *end* field are used for a marking algorithm described in Section 4.6.

The advantage of using the IP option field is listed as follows.

Backward compatibility

It does not disturb ICMP.

No reassemble process of fragmented packets

The computation complexity is reduced to $O(\sum_d S_d)$.

Capability of distinguishing marked packets

The efficiency of collecting packet at a reflector or a victim is facilitated.

A problem pointed out by Snoeren is that each byte of additional overhead reduces system bandwidth by approximately 1% in an average packet size of approximately 128-byte[15]. Although our scheme adds 12-byte per packet, not all the packets need to be marked. This indicates the bandwidth consumption can be controlled by the probability p. For example, if assumed the average distance d to the origin is 16 (this is median value measured by Skitter[26]) and the marking probability p is 0.01, the probability of packet marked at any routers is $1 - (1 - p)^d = 0.05$, thus system overhead could be reduced to approximately 0.6%.

Another possible problem is the situation when an attacker generate a packet filled with other IP options and leaving no space to mark. By assumption, we separated path validation issue from traceback problem and only assumes anonymous packets, therefore we forced a mark to be overwritten even if other options exists. One way to avoid this problem is to "reserve" the path as well as RSVP[27], allowing innocent senders to signal each router not to mark. We discuss this more in Section 7.1.

4.6 RPPM Algorithms

Hereunder, we describe the detail of RPPM algorithm consisting of the following three algorithms: marking, reflection, and reconstruction.

4.6.1 Marking Algorithm

Figure 5 describes the marking algorithm. The marking is conducted on a router and reflector. The reflector should execute this marking algorithm after the reflection algorithm described in Section 4.6.2. The basic behavior is nearly equal to conventional PPM.

Since neither source nor destination address of a path from a reflector to victim is forged, it is efficient not to mark any more at the downstream routers from the reflector in this algorithm. Accordingly, further optimization that defines the explicit reflection flag to decide not to mark any more can be considered. However, an attacker can also generate the reflection flags by assumptions; such optimization may disable PPM. From this consideration, the presence of reflector should not change the semantics of marking algorithm. In general, each mark in packet should be overwritten in equal probability even if any extension might be proposed.

```
Marking procedure at Router R and Reflector Rf:
  for each packet $w$
    let $x$ be a random number from [0..1)
    if $x < p$ then
      write $R$ into $w$.start and 0 into $w$.distance
    else
      if $w$.distance = 0 then
        write $R$ into $w$.end
      increment $w$.distance
```

Fig. 5 Marking algorithm.

4.6.2 Reflection Algorithm

Figure 6 describes the reflection algorithm. The reflection is conducted on a reflector, which can be any Internet server. The reflection algorithm has two procedures: *storing* and *reflecting.* These two procedures share statistics of hash table H. The storing procedure slots in an incoming request packet into H keyed by its source address. Identically marked packets are put into the identical hash entry, then the total count of received packets are incremented. To copy the probability distribution of marks, the reflecting procedure marks an outgoing reply packet where its source address and the entry of H matches. Also the reflecting procedure smoothes H by timer T of sampling period.

In this algorithm, it is necessary to distinguish request and reply packets to generate statistics. To separate packets into request and reply, *connection tracking* technique is required. Most existing filtering architectures provide such connection tracking feature, we do not refer it here.

4.6.3 Reconstruction Algorithm

Figure 7 describes the reconstruction algorithm. As we mentioned in Section 4.6.1, skipping reconstruction of the path between a reflector and a victim is efficient. However, the use of explicit flags to decide marking may change the semantics of PPM allowing malicious improper use by attackers. For this reason, we simply infer potential reflectors from received packets. That is, if the source address

```
let $T$ be a timer of sampling period
let $H$ be a hashtable
let mark be tuples(start, end, distance, count), which is an attribute of a packet
let entry in $H$ be tuples (address, marks, total_count)
Storeing procedure at Reflector Rf:
  for each incomming request packet $w$
    if $H$ contains $w$.source then
      let $e$ be an entry($w$.destination) in $H$
      increment $e$.total_count
    if $w$.mark then
      if $H$ contains $w$.source then
        let $e$ be an entry($w$.source) in $H$
        if $e$.mark $=$ $w$.mark
          increment $e$.mark.count
      else
        put entry($w$.source, $w$.mark, 0) into $H$
Reflecting procedure at Reflector Rf:
  for each outgoing reply packet $w$
    if $H$ contains $w$.source then
      let $e$ be an entry($w$.destination) in $H$
      if $T$ expired then
        exp_smoothing(total_count of $e$)
        exp_smoothing(each count of $e$.marks)
        reset $T$
      let $x$ be a random number from [0..$e$.total_count)
      select mark $m$ from $x$ in marking statistics
      if $m \neq$ NULL
        write $m$ into $w$.{start,end,distance}
```

Fig. 6 Reflection algorithm.

and the start field of mark are identical, such packet is reflected by the reflector. Note an attacker may deceive this inference in very low probability, because an attacker can still send a packet with matched pair of source address and start field, and its mark may not be overwritten.

5. Practical Implementation

The practical implementation of RPPM is based on Linux kernel 2.4.x and Netfilter/IPtables [28]. Linux has a 25 percent share in the operating system software market [29], thus our implementation can be deployed easily. Netfilter/IPtables is a framework for packet mangling outside the normal Berkeley Socket interface.

To control a packet mangling operation, a

```
Path reconstruction procedure at vicitim $V$:
  let $G$ be a tree with root $V$
  let edges in $G$ be tuples (start, end, distance)
  let $H$ be a hashtable
  let entries in $H$ be tuples (reflector, distance)
  for each packet $w$ from attacker
    if $w$.start $=$ $w$.source then
      put entry($w$.start, $w$.distance) into $H$
    if $w$.distance $= 0$ then
      insert edge($w$.start, $V$, 0) into $G$
    else
      insert edge($w$.start, $w$.end, $w$.distance) into $G$
  remove any edge$(x, y, d)$
    with $d \neq$ distance from $x$ to $V$ in $G$
  extract path($R_i..Rj$)
    by enumeratin acyclic path in $G$
  show all entries(reflector, distance) in $H$
```

Fig. 7 Reconstruction algorithm.

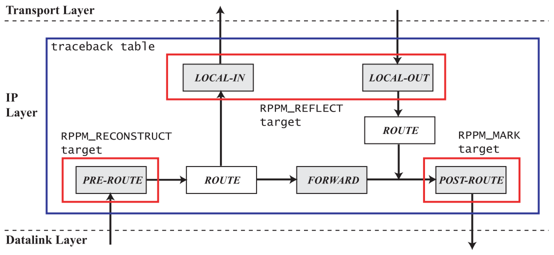

Fig. 8 Implementation of RPPM.

set of hook points called *table* can be defined. We select the following four hook points as `traceback` table in IPv4 layer: `PRE_ROUTING` (after IP checksum calculation, before routing), `LOCAL_IN` (when an incoming packet is destined for a local process), `LOCAL_OUT` (when an outgoing packet is created locally), and `POST_ROUTING` (any outgoing packet after routing).

In each hook point on the table, a packet mangling operation called *target* can be registered. We provide three targets to execute marking, reflection, and reconstruction algorithms: `RPPM_MARK`, `RPPM_REFLCT`, and `RPPM_RECONSTRUCT` targets. **Figure 8** locates the role of these targets and the working hook points. The detail of each targets are explained in the rest of this section.

5.1 Marking

The marking algorithm is implemented by the `RPPM_MARK` target. Since this module works

```
root# iptables -t traceback -A POSTROUTING \
      -j RPPM_MARK --prob <probability>
```

Fig. 9 Using the RPPM_MARK target.

```
root# iptables -t traceback -A INPUT     \
      -j RPPM_REFLECT --period <seconds> \
      --smooth <smoothing constant>
root# iptables -t traceback -A OUTPUT    \
      -j RPPM_REFLECT
```

Fig. 10 Using the RPPM_REFLECT target.

on POST_ROUTING, the packets sent from local processes and those forwarded from other interfaces can be marked equally. **Figure 9** shows the command to use this module. In addition, it can automatically mark the particular interface address into a packet.

5.2 Reflection

The reflection algorithm is implemented by the RPPM_REFLCT target. **Figure 10** shows the command to use this module. To execute the storing procedure for statistics, it works at hook point LOCAL_IN with the smoothing constant and the sampling period of statistics. The range of smooth is between 0 and 100, the recommended argument is 30. The other argument period is specified in seconds, the recommended value can be led from Formula (2) described in Section 4.4.

5.3 Reconstruction

The reconstruction algorithm is implemented by the RPPM_RECONSTRUCT target. **Figure 11** shows the command to use this module. The argument period is specified in seconds, being used to reset the collection of packets.

This module shows the list of potential attack graph rooted by a victim with depth first search as well as collects packets. Current implementation provides this information through the Linux proc file system[30], because packets are collected in kernel space and do not need to copy all of them into user space, only the result of attack graph is enough. **Figure 12** demonstrates a result of a reconstructed path list by opening /proc/net/rppm_reconstructed_path.

5.4 Applying RPPM

We arranged the applying RPPM at routers, reflectors, and victims as follows.

At routers

The probabilistic packet marking should be conducted by RPPM_MARK target shown in Fig. 9. The probability of packet marking is recommended to deploy identically among the routers.

```
root# iptables -t traceback -A PREROUTING \
      -j RPPM_RECONSTRUCT --period <seconds>
```

Fig. 11 Using the RPPM_RECONSTRUCT target.

```
root# cat /proc/net/rppm_reconstructed_path
[0]192.168.1.2 [1]192.168.1.1 [2]192.168.2.2
[0]192.168.1.3 [1]192.168.1.1 [2]192.168.2.3
Potential reflector: 192.168.1.2 (d=0)
Potential reflector: 192.168.1.3 (d=0)
```

Fig. 12 List of reconstructed paths with two potential reflectors.

At reflectors

The probabilistic mark reflection should be conducted by RPPM_REFLECT target shown in Fig. 10. After reflection, the probabilistic packet marking should also be conducted by RPPM_MARK target shown in Fig. 9, in order to tell their own addresses to a victim.

At victims

The reconstruction process should be conducted by RPPM_RECONSTRUCT target shown in Fig. 11. To show the list of reconstructed path, open file as shown in Fig. 11.

6. Evaluation

We conducted several evaluations to investigate the effectiveness of RPPM. The first evaluation examines the effect of reflection algorithm. The second evaluation investigates the performance of our prototype implementation. Finally, the qualitative comparison of existing similar schemes and RPPM scheme is discussed.

6.1 Effect of RPPM Reflection

The effect of the reflection algorithm should not violate the semantics of PPM. In other words, the necessary number of packets to reconstruct paths at a victim should not be changed by the presence of reflection. To prove this ideal characteristic, we have evaluated the number of packets required to reconstruct paths of various distance d from 1 to 32 (it is more than almost all Internet paths[13),26)]), over 1,000 random test runs for each d before and after reflection. In this evaluation, we connected an attacker A, router R, reflector Rf, and vic-

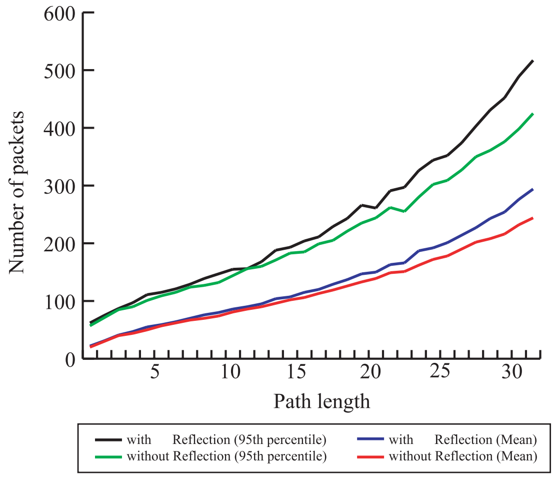

Fig. 13 Number of packets for path reconstruction, with and without a reflection.

Table 1 Performance of internal operations (M: mean, SD: standard deviation).

Operations	M (μsec)	SD (μsec)	smpls
`RPPM_MARK`			
write a mark	5.47	0.656	20
overwrite a mark	1.04	0.292	30
add an end of edge	0.837	0.335	51
increment distance	0.916	0.297	482
do nothing	0.477	0.135	417
All	0.833	0.762	1000
`RPPM_REFLECT`(input)			
increment count	2.22	0.609	456
add a new entry	4.21	0.854	16
update an entry	4.13	0.803	528
All	3.29	1.19	1000
`RPPM_REFLECT`(ouput)			
decide not to mark	570	550	396
decide to mark	618	558	604
All	599	555	1000
`RPPM_RECONSTRUCT`			
collect a packet	2.25	0.587	1000
reconstruct a path	372	20.6	1000

tim V with 100 BaseTX closed network. Each hosts was applied particular RPPM modules on Linux kernel 2.4.12 and Netfilter/IPtables 2.4 with a marking probability $p = 0.05$. Reflection was conducted after packets traversed d-hop marking. We assumed flooding rate F as 1000 per second and necessary precision number P as 2, and sampling period T as 40 seconds (because d was at most 32).

In **Fig. 13**, we graph the mean and 95th percentile of packet counts both before and after reflection. As shown in this result, each situation of mean and 95th are approximately the same. Consequently, it indicates that by applying RPPM modules the presence of reflection does not violate the semantics of PPM.

In addition, most paths can be resolved with between one and two hundred, and even the longest paths can be resolved with less than six hundreds. This result shows that our scheme significantly reduces the number of packets for reconstruction compared with the preceding PPM schemes [6),11)]

6.2 Performance

To investigate the overhead of marking, reflection, and reconstruction algorithms, we measured the performance of internal operation in each function. For this experiment, we prepared two hosts (Celeron 400 MHz and 128 Mbyte of RAM) connected by 100 BaseTX closed network, and assumed a marking probability p is 0.05, distance d to the origin is 16. In this environment, we sent a burst of 1,000 packets and measured performance of each operation. In the evaluation of reflection procedure, we assumed flooding rate F as 1000 per second and necessary precision number P as 2, and sampling period T as 15 seconds (because d was 16). The result of mean and standard division are shown in **Table 1**.

In marking procedure, writing a new mark consumes five times longer (5 microseconds in this case) than other marking operations. This overhead is caused by current Linux IP implementation, which requires a copy of the packet to write the new IP option. We expect that IP options becoming more popular, the optimization of processing IP options will be enhanced.

With regard to the reflecting procedure, especially the operations that reflect a mark into an outgoing packet consumes more than hundred microseconds. Although the memory utilization simply depends on the number of routers, we allocated same memory size of hash space for each distance because of a simple implementation. As a result, the further distance increases the number of routers in the attack path, thereby the hash space for closer distance are thinly used. In this situation, the reflecting procedure scans both used and unused memory space in order to search the mark where total index and random index match. This inefficiency linearly rebounded to the evaluation. For further optimization, closer distance hash space should be reduced by preparing different hash functions for each distance. The result of the optimized memory allocation will decrease the unused space exponentially, thus overhead will be reduced.

Table 2 Comparing with other packet marking schemes.

	PPM	**APPM**	**RPPM**
Reflector attacks ready	No	No	Yes
Distributed attacks ready	No	Yes	Yes
Managing cost	Low	High	Low
Reconstruction complexity	$O(\sum_d S_d \Psi_d^k)$	$O(\sum_d S_d \Psi_d)$	$O(\sum_d S_d)$
Encoding a mark into	IP Identification field	IP Identification field	IP option field
Additional data size	0-byte	0-byte	12-byte
ICMP-compatibility	No	No	Yes
Practical implementation	No	No	Yes

6.3 Qualitative Comparison

To compare characteristics with similar schemes and RPPM, we picked up two preceding schemes that adopt the end-host approach: conventional PPM scheme[6)] and the advanced probabilistic packet marking (APPM) scheme[11)]. The result of qualitative comparison is arranged in **Table 2**. The most apparent difference is the readiness for reflector attacks.

While PPM scheme constrained its design with single source attacks due to success the reduction of each mark, the reconstruction complexity is $O(\sum_d S_d \Psi_d^k)$ for DDoS attacks. On the other hand, APPM dealt with multiple source attacks by using a large set of hash functions and also retains the use of the IP Identification field, that reduced the reconstruction complexity to $O(\sum_d S_d \Psi_d)$. However, APPM assumed the upstream router map, which increased the managing cost for reflector attacks. Accordingly, preceding schemes were unwilling to discard the encoding into IP Identification field, PPM sacrificed the readiness of multiple source attacks and APPM increased the managing cost.

In contrast, we have encoded a mark into the IP option fields in order to deal with multiple source attacks. At the same time, we do not require any additional information such as network map, the managing cost is minimal. Furthermore, a result of using the IP option field, we ensured ICMP-compatibility. Although the overhead of system may be increased by the use of the IP option field, the overhead can be controlled by possibility p and most routers become powerful enough to process IP options, we believe that our scheme is feasible.

Finally, whereas PPM and APPM only showed simulation results, we have implemented practically.

7. Future Work

Yet there are still a number of limitations, we state two main future research topics in this section: path validation and backward compatibility.

7.1 Path Validation

An attacker may gain access to routers along the path from attacker to victim. Furthermore, the attacker may control the routers to let them subvert the marking. The victim cannot differentiate between such malicious packets and genuine marked packets.

In order to validate the reconstructed attack graph, one possible solution is to share the key among trusted routers and encrypt the mark. There are two approaches for sharing the key: an *overlay network* approach and *path reservation* approach. The overlay network approach is attempted by Centertrack[14)], Centertrack has a potential to create the network among trusted routers in order to share the key. The other, the path reservation approach has not yet explored, however existing architecture such as RSVP[27)] has a potential of key distribution platform. In addition, either the overlay network or path reservation approach can apply multiparty protocol such as simplified VSS[31)], which enables to distribute the key into n routers, and decode even if m routers are malicious.

7.2 Backward Compatibility

Currently, our design and implementation focused on IPv4 and ensured ICMP-compatibility. However, there are still different architectures such as IPv6, IP encapsulation (e.g., IPsec, tunneling, etc.). While we do not attempt to propose a complicated scheme also being dealt with such architectures, we believe that similar method can be applied.

8. Conclusion

In this paper, we presented the design, implementation and evaluation of RPPM scheme, which is a traceback scheme against DDoS attacks including traffic laundered by reflectors. Our main contribution is that we have achieved traceback beyond the *reflector attack* which is one of the hardest DDoS attacks in today's Internet. We have extended PPM to render reflectors ineffectual by reflecting marking statistics of incoming packets at reflectors. As well, we have encoded a mark into the IP option field to ensure ICMP-compatibility, whereas preceding schemes dismissed it. Thus, RPPM can traceback beyond reflectors, ensures ICMP-compatibility, and eliminates possibility of failure in attack path reconstruction. Simulation results have shown RPPM retains the semantics of conventional PPM on a path between an attacker and a reflector. Also, its performance is almost feasible for practice, and we have shown the direction of optimization. We will develop our practical implementation further by open source in the near future. Although there are two main future research topics, path validation and backward compatibility, we believe our scheme will be one of the best solutions towards an automated widely deployed traceback system.

References

1) Houle, K.J. and Weaver, G.M.: *Trends in Denial of Service Attack Technology*, Computer Emergency Response Team (2001). `http://www.cert.org/archive/pdf/DoS_trends.pdf`
2) Moore, D., Voelker, G.M. and Savage, S.: Inferring Internet Denial-of-Service Activity, *USENIX Security Symposium* (2001).
3) Senie, D.: *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing*, RFC2827 (2000).
4) Computer Emergency Response Team: *CERT Advisory CA-2000-01 Denial-of-Service Developments* (2000). `http://www.cert.org/advisories/CA-2000-01.html`
5) Paxon, V.: An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks, *ACM Computer Communication Review*, Vol.31 (2001).
6) Savage, S., Wetherall, D., Karlin, A. and Anderson, T.: Practical Network Support for IP Traceback, *ACM SIGCOMM 2000*, pp.295–306 (2000).
7) Asta Networks Inc. `http://www.astanetworks.com/`
8) Burch, H. and Cheswick, B.: Tracing anonymous packets to their approximate source, *USENIX Systems Administration Conference (LISA)*, pp.319–327 (2000).
9) Bellovin, S.M.: *ICMP Traceback Messages* (2000). Internet Draft: draft-bellovin-itrace-00.txt
10) Barros, C.: *A Proposal for ICMP Traceback Messages* (2000). Internet Draft: draft-cesarb-itrace-01.txt
11) Song, D. and Perrig, A.: Advanced and Authenticated Marking Schemes for IP Traceback, *IEEE INFOCOM 2001* (2001).
12) Cheswick, B. and Burch, H.: The Internet Mapping Project. `http://cm.bell-labs.com/who/ches/map/index.html`
13) Govindan, R. and Tangmunarunkit, H.: Heuristics for Internet Map Discovery, *IEEE INFOCOM 2001*, pp.1371–1380 (2001).
14) Stone, R.: Centertrack: An IP Overlay Network for Tracking DoS Floods, *USENIX Security Symposium* (2000).
15) Snoeren, A.C., Partridge, C., Sanchez, L.A. and Jones, C.E.: Hash-Based IP Traceback, *ACM SIGCOMM 2001* (2001).
16) Schnackenberg, D., Djahandari, K. and Strene, D.: Infrastructure for intusion detection and response, *First DARPA Information Survivability Conference and Exposition* (2000).
17) Snort. `http://www.snort.org/`
18) Zhang, Y. and Paxon, V.: Detecting Stepping Stone, *USENIX Security Symposium* (2000).
19) Computer Emergency Response Team: *CERT Advisory CA-96.26 Denial-of-Service via pings* (1996). `http://www.cert.org/advisories/CA-96.26.ping.html`
20) Paxon, V.: End-to-End Routing Behaivor in the Internet, *IEEE/ACM Transactions on Networking*, Vol.5, No.5, pp.601–605 (1997).
21) Stoica, I. and Zhang, H.: Providing guaranteed services without per flow mangagement, *ACM SIGCOMM 1999*, pp.89–94 (1999).
22) Jacobson, V.: Congestion Avoidance and Control, *ACM Computer Communication Review*, Vol.18, No.4, pp.314–329 (1988).
23) Everette, G. and Jr, S.: Exponential smoothing: The state of the art, *Journal of Forecasting*, Vol.4, No.1, pp.1–38 (1985).
24) Bowerman, B.L. and O'Connell, R.T.: *Time Series and Forecasting*, Duxbury Press, North Scituate, Massachusetts (1979).
25) Postel, J.: *Internet Protocol*, RFC791 (1981).
26) Cooperative Association for Internet Data Analysis: Skitter Analysis. `http://www.caida.org/tools/measurement/skitter/`
27) Braden, R. and Zhang, L.: *Resource ReSerVation Protocol*, RFC2209 (1997).
28) The Netfilter/IPtables project. `http://netfilter.samba.org/`
29) International Data Corp. `http://www.idc.com/`
30) Faulkner, R. and Gomes, R.: The Process File System and Process Model in UNIX Sys-

tem V, *USENIX Association Winter Conference* (1991).

31) Gennaro, R., Rabin, M.O. and Rabin, T.: Simplified VSS and Fast-track Multiparty Computations with Applications to Threshold Cryptography, *ACM PODC '98* (1998).

(Received November 29, 2002)
(Accepted June 3, 2003)

Nobuhiko Nishio was born in 1962. He received his B.E. and M.S. degree from the University of Tokyo in 1986 and 1988, and Ph.D. degree from Keio University in 2000. He is currently an associate professor at Ritsumeikan University BKC and a researcher at PRESTO of JST. Since 1993 till 2003 he had worked at Keio University SFC and got Yamashita memorial research award in 1994. His current research interests are ubiquitous computing and wireless ad-hoc sensor networks. He is a member of the IPSJ, the IEEE-CS and the ACM.

Noriyuki Harashima has received his B.S. and M.S. degree from Keio University in 2000 and 2002 respectively. Receiving his M.S., he is currently working for ACCESS Co., LTD. in engaged in research and develepement of Java related system software. His research interest is secure ubiquitous computing especially for IDS and traceback technology for distributed DoS attack in the Internet.

Hideyuki Tokuda was born in 1952. He received his B.S. and M.S. degrees from Keio University, in 1975 and 1977 and Ph.D. degree in Computer Science from the University of Waterloo in 1983. He is currently a Professor in the Faculty of Environmental Information, Keio University. His research interests include real-time systems, multimedia systems, mobile systems, communication protocols, massively parallel/distributed systems and embedded systems. His current research interests are ubiquitous computing and networking systems. He is a member of the IEEE, the ACM, the IPSJ and JSSST. He is currently the chair of the SIGUBI in IPSJ.