An IP Traceback Technique against Denial-of-Service Attacks

Zhaole Chen, Moon-Chuen Lee
Computer Science & Engineering Department, the Chinese University of Hong Kong
{*zlchen, mclee*}*@cse.cuhk.edu.hk*

Abstract

Reflector attack [9] belongs to one of the most serious types of Denial-of-Service (DoS) attacks, which can hardly be traced by contemporary traceback techniques, since the marked information written by any routers between the attacker and the reflectors will be lost in the replied packets from the reflectors. We propose in this paper a reflective algebraic marking scheme for tracing DoS and DDoS attacks, as well as reflector attacks. The proposed marking scheme contains three algorithms, namely the marking, reflection and reconstruction algorithms, which have been well tested through extensive simulation experiments. The results show that the marking scheme can achieve a high performance in tracing the sources of the potential attack packets. In addition, it produces negligible false positives; whereas other current methods usually produce a certain amount of false positives.

1. Introduction

A Denial-of-Service (DoS) attack is characterized by an explicit attempt by an attacker to prevent legitimate users of a service from using the desired resources [10]. While launching their attacks, the attackers normally generate a huge volume of packets directed to the target systems known as victims, causing a network traffic congestion problem. Thus the legitimate users would be prevented from gaining access to the systems being attacked. This paper focuses on using an innovative marking scheme to defend against DoS attacks. We propose a methodology, based on a packet marking technique, to trace DoS attacks, especially reflector attacks.

Reflector attacks belong to the category of the most serious DoS attacks. Unlike other DoS attacks, the number of attack packets sent out by a reflector attacker would be amplified many times, flooding the victim's network. The attack packets reaching the victim are not directly from the attacker; they are actually generated by some hosts known as reflectors. When such reflectors receive the packets from a reflector attack, they would create many times more packets with a destination address of the victim. A detail description of reflector attack is presented in section 2. This type of attack is more difficult to trace and almost all previous traceback techniques cannot handle it.

The rest of this paper is organized as follows. We introduce reflector attacks and related traceback techniques proposed in the literature in section 2 and section 3 respectively. Section 4 describes our traceback algorithm. Section 5 analyses the backward compatibility of our marking scheme. After showing the experiment results in section 6, we conclude this paper in section 7.

2. Reflector attacks

A reflector is any IP host that will return a packet if sent a packet [9]. A reflector attack is an indirect attack in that intermediary nodes (routers and various servers), better known as *reflectors*, are innocently used as attack launchers [11]. Some major reflector attacks such as smurfing, SYN flooding, RST flooding, ICMP flooding and DNS reply flooding are summarized in [11]. We use smurfing, a typical reflector attack, as an example to introduce how a reflector attack is launched.

With IP spoofing as an intermediate step, the launch of a smurfing attack involves spoofing a number of ICMP echo (ping) packets with the victim's IP address as the source address and a directed broadcast address as the destination. This kind of attack can consume a lot of network and host resources with relatively few spoofed packets. There are three major components constituting a reflector attack—the attacker, the amplifying subnet (i.e. reflectors), and the victim. The attacker sends ICMP echo packets with the victim's IP address as the source address to the broadcast address of an amplifying network as the destination address. So the packets appear to have been sent by the victim. Since they are sent to a broadcast address of a local network, all the hosts, except those whose configuration has been specified not to respond to ICMP broadcast packets, in the local network will respond to each of the packets. Therefore, smurf is a kind of amplified DoS attack. Because of this amplifying effect, an individual reflector attacker can send the packets at a much lower rate compared to the packet rates created by ordinary DoS attackers who flood the victim directly. In practice, a smurf attacker can first compromise a set of hosts called slaves, and then it instructs each slave to send ICMP echo packets to a number of amplifying networks.

Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC 2003)
1063-9527/03 $17.00 © 2003 **IEEE**

Therefore, the attack can be considered as a kind of distributed DoS attack.

There is another type of reflector attacks which doesn't exploit the amplifying power of subnets. The attacker will locate a large number of reflectors to launch its attack. Therefore, the attacker has to send request packets to each of the reflectors. As long as the number of reflectors is large enough, the attacker can flood the victim. In a direct ICMP ping flooding attack, around 5000 reflectors are needed to flood a victim's T1 link if each reflector sends a query every second [11]. This category of reflector attacks is more difficult to trace than the "amplifying reflector attacks" since the traffic volume from each link may be smaller, however, such attacks are much more difficult to launch. Therefore, in this paper we focus on the reflector attacks which exploit the amplifying effect of subnets.

There are some traditional solutions to defend against the reflector attacks. For example, to defend against smurfing attack: One solution is disabling the translation of layer 3 broadcasts into layer 2 broadcasts at the border router; that is, the router will filter any packet with a broadcast source address. However this will disable many useful services such as ARP, audio sharing and so on. Another solution is configuring the host not to respond to broadcast ping packets or to ignore all ping packets. This will result in the loss of the ICMP functionality. So we strongly recommend using IP traceback technique to identify the attackers, and then take other measures to stop the attacks.

While dealing with these types of DoS attacks, one serious problem of the traditional marking schemes is that the information marked by the routers between the attacker and the reflectors is lost when the reflectors have sent the "reply" packets to the victim after processing the attack packets at application layer. Thus the existing marking schemes cannot trace part of the attack path, namely the sub-path from the reflector to the attacker.

3. Related work

Generally speaking, there are two major categories of traceback techniques—one is to trace a single packet, and the other is to use a large number of packets for tracing the attacker. Hash-based traceback [7], based on the former technique, digests and logs some particular information of each packet on the routers. The victim can query its upstream routers whether a certain packet has passed through them. This method has two drawbacks: it requires a large-scale database on each router to store and manage the packets information; the queries must be done before the relevant records in the database have been updated. The scheme proposed in this paper belongs to the latter technique. Different IP traceback methods, based on using a large number of packets, have been proposed in the literature; examples include link testing, ICMP traceback, and some packet marking based methods. However, all of these methods cannot trace the source of a reflector attack.

Dean and Franklin proposed an algebraic marking scheme for IP traceback [1], which is based on the mathematical theory of *Linear Algebra* and *coding theory*. The main drawback of this algebraic marking scheme is that it is not effective in dealing with multiple attacks. In addition, it cannot traceback reflector attacks. Motivated by Dean's idea, we propose in this paper a Reflective Algebraic Marking Scheme, which can effectively deal with DoS attacks including reflector attacks. Our approach has low network and router overheads similar to the algebraic marking scheme; in addition, it can efficiently perform IP traceback even in the presence of multiple attacks including reflector attacks.

4. Reflective marking scheme

In this section, we introduce our reflective algebraic marking scheme in detail. To provide the necessary background for the presentation of the proposed method, we give some definitions first, and then the basic assumptions. Many of the definitions and assumptions follow the work of Savage *et al.* [2].

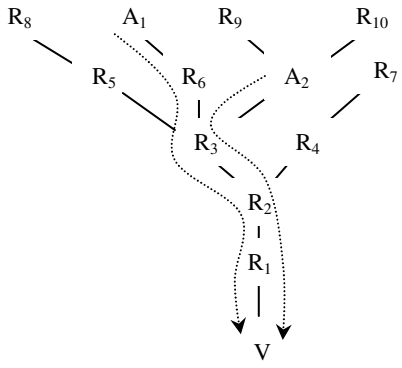

Figure 1. An upstream routers map as seen from the victim. There are two attack paths indicated by the dotted lines.

4.1. Definitions

Figure 1 depicts an upstream routers map from the view of the victim. We use *V*, *R*, and *A* to denote the victim, router, and attacker respectively. An upstream routers map is a map describing the topology of the upstream routers of a single host. The upstream routers map contains the IP addresses of the upstream routers. Here upstream is used to describe routers viewed from the victim. For example, R_9 and R_{10} are the upstream routers of A_2. In this graph, there are two attack paths that are represented by the dotted lines, one is (A_1 R_6 R_3 R_2 R_1),

and the other is (A_2 R_3 R_2 R_1). The distance between two hosts means the number of routers in the attack path between them. For example, in the attack path (A_1 R_6 R_3 R_2 R_1), the distance between router R_6 and the victim is 3. As we know, some routers might be compromised by the attacker and so the marked information in the packets may be forged. Thus we limit the traceback problem to finding a candidate attack path that contains a suffix of the real attack path, and such suffix is called a valid suffix of that path. For example, path (R_6 R_3 R_2 R_1) is a valid suffix of the real attack path (A_1 R_6 R_3 R_2 R_1). We say a traceback scheme is robust if the attackers cannot prevent the victim from finding the candidate paths containing the valid suffixes of the attack paths. We call a router false positive if it is in the reconstructed attack path and not in the real attack path.

4.2. Assumptions

In order to make our marking scheme more practical and effective, the following assumptions have been made in devising our proposed IP traceback method.

1) Attackers are able to generate any packets
2) Multiple attack paths may exist
3) Packets may be reordered or lost
4) Routes between the attacker and the victim are fairly stable
5) The resources of the routers are limited so that the routers cannot perform too much processing per packet.
6) Attackers might be aware that they are being traced
7) Attackers may send a large number of packets
8) Routers might be compromised; but the nearest routers should not be compromised in big proportion.

The first five assumptions are quite easy to understand with the knowledge of current network infrastructure. The sixth one is a conservative evaluation of the abilities of the attackers. The sophisticated attackers should be aware that they are being traced and may send fake packets to make the victim confused. So the traceback method proposed must consider such an ability of the attackers. Like the probabilistic marking scheme [2], our marking scheme marks packets with a very low probability; so it requires a certain number of packets, sent by the attacker, to reconstruct the attack paths. If some routers are compromised, we could only reconstruct the attack paths to the corresponding relevant comprised routers since a compromised router could tamper the information marked by its upstream routers. Therefore, we will use a *valid suffix* instead of the entire attack path to assess the robust of a traceback technique. One thing to remind is that the nearest routers should not be compromised; otherwise they can tamper any messages the upstream routers have marked, so that the victim might reconstruct totally wrong paths.

4.3. Reflective Algebraic Marking Scheme

Given the basic definitions and assumptions, we describe our specific marking scheme now. At first, we introduce the basic mathematical theory, and then the marking, reflection and reconstruction algorithm.

4.3.1. Basic mathematical theory:

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & x_n^2 & \dots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} A_1 \\ A_2 \\ \dots \\ A_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix}$$

The above equation is a matrix equation (or system of equations) with *Vandermonde* matrix coefficient. In *Linear Algebra*, there is a theorem stating that the above matrix equation, with A_i's unknown, has a unique solution if and only if the x_i's are distinct [4]. By applying *field theory* to the above theorem, we can obtain a similar theorem over GF(p), where GF denotes *Galios Field* and p is a prime number if the x_i's and y_i's are elements in GF(p) [8].

4.3.2. Reflective algebraic marking scheme

Our reflective algebraic marking scheme consists of three algorithms: marking, reflection, and reconstruction, deployed on the routers, the reflectors and victim respectively. The reflection algorithm is employed to handle reflector attacks.

The information recorded in each marked packet includes three integer values in the IP header: x, *distance* and *Fullpath*; where x is a packet related value; *distance* is the distance between the router which first marks this packet and the victim. In light of the technique of Dean *et al.*[1] for reducing the value of *Fullpath*, we split the IP address of a router R_i into c identical fragments, and use $A_{i,j}$ (j = 1, 2, … , c) to denote the value of each fragment. For example, if router R_1's IP is 137.189.89.101 and we split it into 4 (c = 4) chunks, then $A_{1,1}$ = 137, $A_{1,2}$ = 189, $A_{1,3}$ = 89, $A_{1,4}$ = 101. And the value of *Fullpath* can be computed as follows:

$$Fullpath = (A_{1,1} + A_{1,2}x + A_{1,3}x^2 + A_{1,4}x^3 + A_{2,1}x^4 + A_{2,2}x^5 + A_{2,3}x^6 + A_{2,4}x^7) \bmod p,$$

where $A_{i,j}$'s (i =1, 2; j = 1, 2, 3, 4) form the IP addresses of two adjacent routers, and p is the smallest prime number larger than 255 (=2^8 – 1), i.e. 257. If the router is adjacent to the victim, the last 4 terms of *Fullpath* should be omitted. The purpose of "**mod** p" in the above

formulation is to reduce the value of *Fullpath* so that it would occupy fewer bits in the IP header. Considering the bits needed to store the *Fullpath* value and the attack paths reconstruction time, letting c equal to 4 is an eclectic choice. Figure 2 depicts the marking algorithm for $c = 4$ and we also use $c = 4$ in the following sections.

Marking procedure in router R

```
for each packet P {
   generate a random number u [0, 1) ;
      if (u ≤ q ) {
         /* q is the marking probability adopted by all the routers
         and the victim */
         P.distance = 0;
         randomly select an integer x from 0 to 7;
         P.x = x ;
         Fullpath = (A1,1 + A1,2x + A1,3x^2 + A1,4x^3) mod p;
         // each packet P is associated with a value x
      }
      else {
         if (P.distance == 0) {
            Fullpath = (Fullpath + A1,1x^4 + A1,2x^5
                        + A1,3x^6 + A1,4x^7) mod p;
            // x is recorded in the packet by an upstream router
            P.distance = P.distance + 1;
         }
         else if (P.distance > 0) P.distance = P.distance + 1;
               else call error_handler;
      }
}
```

Figure 2. Marking algorithm.

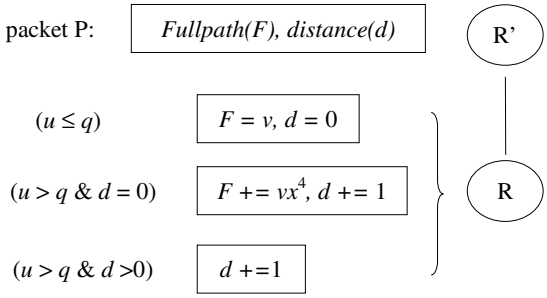

Figure 3. Marking illustration. F and d denote *Fullpath* and *distance* respectively, $v = A_{1,1} + A_{1,2}x + A_{1,3}x^2 + A_{1,4}x^3$, R' is an upstream router of R.

Figure 3 illustrates the marking procedure; F and d denote *Fullpath* and *distance* respectively; v represents the value of $A_{1,1} + A_{1,2}x + A_{1,3}x^2 + A_{1,4}x^3$, where $A_{1,i}$'s (i = 1, 2, 3, 4) are the 4 fragments of the relevant router's IP address. When router R receives a packet from its upstream router R', it first generates a random number u and does the marking depending on u and the distance d recorded in the packet. If $u \le q$, then let $F = v$, $d = 0$; if $u>q$ & $d = 0$, then let $F = F + vx^4$, $d = d + 1$; if $u > q$ & $d > 0$, then increment the distance d by 1. To make the presentation concise in the diagram, we have omitted applying "mod p" to F here.

To resolve the information loss problem caused by reflection, we copy the marked information in each incoming request packet to the outgoing reply packet; this operation is carried out through the reflection procedure by each reflector. Note that the number of request packets and the number of reply packets are asymmetric. For example, the number of packets in a GET request message of FTP is small, but those in the reply message may be large. For this reason, a simple copy operation for the marked information may not work. One possible method is to use a table to store the marked information; the reflector simply collects the marked information in the table and copies the relevant marked information to outgoing packets.

Reflection algorithm:

```
let H be a hash table;
let mark be the tuple(Fullpath, distance, x);
//mark stands for marked information in a packet
let entry in H be a tuple(address, mark, count);
/* address is for storing IP address and count is an integer
initiated as 0, H is sorted by address, then distance, and then x */
Storing marked information at Reflector Rf:
for each incoming request packet w
   if H doesn't contain (w.source, w.mark, _)
      insert into H (w.source, w.mark.distance, 0);
      // source is for storing the source IP address of packet w
Copy operation at Reflector Rf:
for each outgoing reply packet w
   if H contains (w.destination, _, _) {
      select an entry e of the form (w.destination,_,_) in H
        whose count is the smallest and distance is the smallest;
      write e.mark into w.mark;
      increase e.count by 1;
      if (count == bound) delete e1;
      // bound = ln(32)/(p(1-p)^31)
   }
```

Figure 4. Reflection algorithm.

Figure 4 outlines the reflection algorithm. It consists of two procedures for storing marked information and copying marked information to reply packets. These two procedures share statistics of a hash table H. The storing procedure keeps an incoming request packet's marked information in the table H with the source address as a key. Packets with identical marked information are stored only once. The copy procedure copies the marked information to an outgoing reply packet from the record in H whose address matches the packet's destination address. The copy operation is carried out only when the count of the record to be copied is less than a certain upper bound,

since the number of packets X required for the victim to reconstruct a path of length d has the following bounded expectation [2]:

$$E(X) < \frac{\ln(d)}{p(1-p)^{d-1}}.$$

As analyzed in section V, we can set the maximum value of d as 32. Therefore, the upper bound can be set as:

$$bound = \frac{\ln(32)}{p(1-p)^{31}}.$$

We also set an expiration time parameter T for the maintenance of table H. If all records of a certain IP address have not been visited in the past period T, it can be assumed that all the relevant packets replied to that address have been sent out and we therefore delete them. The value of T shall be set according to the configuration of specific reflector.

In this algorithm, it is necessary to distinguish request and reply packets, which requires *connection tracking* technique [12]. Most existing filtering architectures provide such connection tracking feature, which will not be elaborated here.

When 8 (or 4) packets with distinct x's arrive at the victim, the victim can solve the relevant matrix equation in section 4.3.1 to obtain the IP addresses (or address) of two adjacent routers (or the nearest router to the victim) in the attack path. Therefore, we use a set of 8 distinct x's (0-7) to do the marking. The mandatory increment of the *distance* field is used to avoid spoofing by an attacker, thus ensures the *robustness* of our scheme. For attack paths reconstruction, we use a method similar to edge sampling [7] to reconstruct the attack path hop by hop. However, we are not sure about how to group the packets coming from the same path. It will certainly involve a high computation if we check all possible combinations of the marked packets similar to the probabilistic marking scheme [7]. Therefore, we resort to using an upstream routers map of the victim to simplify paths reconstruction. As pointed out by Song and Perrig, it is quite easy to obtain and maintain such an upstream routers map [9]. After receiving enough packets, the victim can reconstruct all the attack paths by the algorithm as outlined in Figure 5.

Reconstruction algorithm

let M denote the upstream routers map;
let G denote the reconstructed attack paths graph and be initialized with only one node V for the victim;
group all marked packets by *distance*;
group each set of packets by x values;
let P_d denote the set of packets marked with distance d ($0 \le d \le$ maxd) and X_k denote the packet subset of P_d with $x = k$;
let *maxd* be the distance from the furthest attacker to the victim;
for every direct upstream router R of V in M {
  count = 0; k = 0;
  while (count <4 and k < 8 and (8-k+count) < 4) {
    /* "8-k+count < 4" implies count will not be 4 after exiting this while loop */
    select a packet from X_k of P_0, which has not been selected before in this *while* loop;
    $path = (A_{1,1} + A_{1,2}x + A_{1,3}x^2 + A_{1,4}x^3)$ **mod** p
    // $A_{1,j}$ (j = 1, 2, 3, 4) form the IP address of R
    // x and *Fullpath* are from the selected packet
    if ($path$ == *Fullpath*) count = count + 1;
    if ((all packets in X_k have been visited)or($path$==*Fullpath*))
      $k = k + 1$;
  }
  if (count == 4) insert R into G next to V;
}
for d = 1 to *maxd*
  for every router R inserted into G in the last loop {
    for every upstream router R' of R in M or R itself {
      k = 0;
      while (k < 8){
        select a packet from X_k of P_d which has not been selected before in this *while* loop;
        $path = (A_{1,1} + A_{1,2}x + A_{1,3}x^2 + A_{1,4}x^3 + A_{2,1}x^4 + A_{2,2}x^5 + A_{2,3}x^6 + A_{2,4}x^7)$ **mod** p
        // $A_{1,j}$ (j = 1, 2, 3, 4) form the IP address of R'
        // $A_{2,j}$ (j = 1, 2, 3, 4) form the IP address of R
        if ($path$ == *Fullpath*) $k = k + 1$;
        if (all packets in X_k have been visited)
          quit this *while* loop;
      }
      if (k == 8) insert R' into G next to R;
    }
  }
output the reconstructed attack paths from graph G

Figure 5. Attack paths reconstruction algorithm.

P_n

	d	F	x
X_0	n	F_{01}	x_0
	…	…	…
	n	F_{0a}	x_0
X_1	n	F_{11}	x_1
	…	…	…
	n	F_{1b}	x_1
…	…	…	…
X_7	n	F_{71}	x_7
	…	…	…
	n	F_{7h}	x_7

Figure 6. Packet set P_n .

The initial stage of the attack paths reconstruction starts from the routers adjacent to the victim. The algorithm first identifies the nearest routers in layer 1 (whose distance to

the victim is 0). The table in Figure 6 depicts the packet set P_n. The routers in layer 1 can be identified using only the packet set P_0 since we have grouped the packets by distance d. For each adjacent upstream router of V in the upstream routers map M, and for each packet in X_i (i = 0, 1, … , 7), the value for *path* is computed as follows (x and *Fullpath* are obtained from the packet):

$$path = (A_{1,1} + A_{1,2}x_i + A_{1,3}x_i^2 + A_{1,4}x_i^3) \bmod p.$$

If *path* is equal to *Fullpath*, we move to another packet set X_{i+1}. If each of the 4 packet sets has at least one packet yielding *path* equal to *Fullpath*, we can conclude that the selected router is on one of the attack paths and insert it into the reconstructed attack paths graph.

Now the victim will identify the routers in other layers after finding the routers in the first layer. Suppose an attack path has been reconstructed from the victim to router R_n in layer n (whose distance to the victim is n-1). Now, we need to identify R_n's upstream router R_{n+1} in layer n+1 by using the packet set P_n. For each router next to R_n in M, and for each packet in X_i (i = 0, 1, … , 7), the value for *path* is computed as follows:

$$path = (A_{1,1} + A_{1,2}x_i + A_{1,3}x_i^2 + A_{1,4}x_i^3 + A_{2,1}x_i^4 + A_{2,2}x_i^5 + A_{2,3}x_i^6 + A_{2,4}x_i^7) \bmod p.$$

If *path* is equal to *Fullpath*, we move to another packet set X_{i+1}. If no packet in X_i yields the same value for *path* and *Fullpath*, we can declare that the selected router is not on the attack paths of this layer (it could be on the paths of other layers). If each of the 8 packet sets has at least one packet yielding the same values, we can conclude that the selected router is on one of the attack paths and insert it into the reconstructed attack paths graph. Here we must check whether a router has marked a single packet twice. This case happens when a request packet has been marked by a router of a reflector and then the reply packet goes through the same router to the victim; so this router could mark the packet again.

By this method, we can try different branches in the upstream routers map to reconstruct all the attack paths. We use 4 packets to identify one nearest router and 8 packets to identify two adjacent routers to make sure that there are no collisions because the corresponding matrix equations have a unique solution.

5. Backward Compatibility

Backward compatibility is the most important issue concerning whether a proposal can be put into practice. As our marking scheme requires writing some information to the IP header of a packet, we should find out those bits in the IP header which can be overwritten.

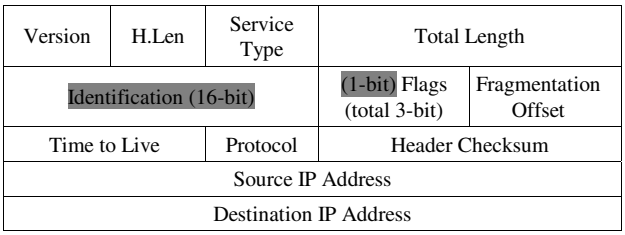

Figure 7. IP header. The shaded fields (17 bits) are little used in current network implementation

Figure 7 shows the structure of the IP header. The 16-bit Identification field allows the destination host to determine which datagram a newly arrived fragment belongs to. Stoica and Zhang pointed out that less than 0.25% of the entire network traffic is fragments [6]; so the bits for the identification field can be overloaded with the marking information. In addition, one out of three bits in the Flags field is of little use in the current version of IP protocol [1]. Thus we can use up to 17 bits to store marking information.

The total number of bits b needed to store the marking information can be estimated as $b = \log_2(p) + \log_2(d) + \log_2(n)$. The first term estimates the bits needed to store *Fullpath*, which has a value less than p. The second term estimates the bits needed to store *distance*, and the third term estimates the bits needed to store x. Letting $c = 4$, $d = 32$, $p = 257$, and $n = 2c = 8$, the above expression for b can be computed to a value no more than 17. The reason for setting $n = 2c$ is that each *Fullpath* is related to $2c$ fragments of two IP addresses; as long as we provide $2c$ distinct values of x, the two IP addresses can be uniquely identified. Thus 3 bits would be needed to store 8 distinct values of x.

There is a tradeoff between the number of packets required for reconstruction and the number of bits needed. A smaller value for c implies a smaller number of packets and a shorter reconstruction time. However, the total number of bits in the IP header that can be used to store the marking information is quite limited, so we eclectically choose $c = 4$ in our implementation. In general, a packet can reach its destination by passing no more than 32 hops [5]. For reflector attacks, we can assume the *distance* field of a packet is less than 64. Therefore, 6 bits would be sufficient for the *distance* field. Then there would be only 8 (17 – 3 – 6) bits left for storing the *Fullpath* value, which ranges from 0 to 256. Thus two of the values will be in collision. In our implementation, if the *Fullpath* value calculated by the router is 256, the router would write 0 to the *Fullpath* field. While doing the attack paths reconstruction, if the *path* value calculated by the victim is 256, the victim would convert it to 0. With this simple technique employed to handle the collision of two different values, the probability of reconstructing a false

positive would be extremely low. In our thousands of attack paths reconstruction experiments, no false positives were generated.

In summary, we have identified sufficient appropriate bits in the IP header for storing the marking information; therefore, the proposed marking scheme is backward compatible with the current version of the IP protocol and can be effectively put into practice.

6. Experiment results

We have performed a good number of simulation experiments to examine the feasibility and evaluate the performance of our marking scheme. The primary objective of the experiments is to investigate some parameters related to the marking scheme: the number of false positives, minimum number of packets needed for attack paths reconstruction, reconstruction time, etc.

In preparation for the simulation experiments, we prepared an upstream routers map with over 200 routers. The routers were then assigned some real IP addresses obtained from the Internet by using the *traceroute* technique. The attack paths are randomly chosen from the paths in the map; and different number of packets are generated and transmitted along each of these paths. Each of the routers simulates marking the packets as defined in the marking algorithm. With the pool of marked packets collected, the victim simulates applying the proposed reconstruction algorithm to reconstruct all the attack paths.

Len1 / Len2	5	10	15	20	25	30
0	660	1070	1550	2080	2710	3190
5	610	1020	1470	1760	2350	2990
10	615	1030	1480	1780	2360	3000
15	620	1035	1490	1790	2370	3020
20	630	1040	1500	1800	2380	3030
25	640	1050	1515	1810	2400	3050
30	645	1055	1530	1825	2410	3060

Len1 / Len2	5	10	15	20	25	30
0	1900	2430	2970	3430	3870	4250
5	1830	2350	2820	3240	3720	4020
10	1840	2360	2830	3250	3730	4030
15	1845	2370	2840	3260	3740	4050
20	1850	2380	2850	3265	3750	4070
25	1860	2390	2860	3270	3760	4090
30	1870	2400	2870	3280	3770	4100

Figure 8: Tables of minimum number of packets needed for reconstruction. Up: q = 0.04; Down: q = 0.01. q: marking probability; Len1: length of the path from attacker to reflector; Len2: length of the path from reflector to victim; nRf: the number of reflectors.

The experimental results show that our proposed marking scheme is feasible and the performance is satisfactory. They also confirm that the attack paths reconstruction algorithm yields negligible false positives.

Figures 8 tabulates the results of the minimum number of packets required to reconstruct paths of varying lengths with 95% probability of success, and the marking probabilities q are 4% and 1% respectively. In a reflector attack, the attack path is composed of two parts, namely the sub-path from the attacker to the reflectors, and the sub-path from the reflectors to the victim. So the total path length would be equal to the sum of the lengths of the two parts. We set the number of reflectors *nRf* involved in the reflector attacks equal to 30. For the two sets of experiments presented in Figures 8, we varied the attack path (from attacker to reflector) length parameter *Len*1 from 5 to 30, and the attack path (from reflector to victim) length parameter *Len*2 from 0 to 30. Each data point in each of the tables corresponds to an average of the data values obtained from over 200 independent experiments.

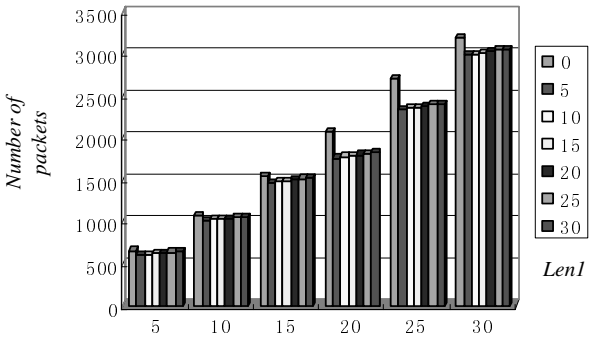

Figure 9: Bar chart representation of minimum number of packets needed for attack paths reconstruction (q = 0.01).

When the parameter *Len2* is 0, the simulated attacks would be the ordinary DoS attacks and not reflector attacks. When compared with FMS [7], and scheme 1 of the advanced marking scheme [9], our marking scheme requires significantly less packets for attack paths reconstruction. Our scheme is also slightly better than scheme 2 with m>7 (for the case with a minimum number of false positives), and not worse than scheme 2 with m>6 (for the case with a second minimum number of false positives) of the advanced marking scheme [9]. In addition, our scheme also outperforms other traceback schemes based on the criterion of producing the least number of false positives.

For the cases with *Len*2 not equal to 0, the simulated attacks belong to reflector attacks. Figure 9 shows a bar chart view of the minimum number of packets needed for attack paths reconstruction with q = 0.01. The bar chart

representation for q = 0.04 should look similar. From Figure 9, it can be observed that the minimum number of packets needed for paths reconstruction depends primarily on the path (from the attacker to the reflector) length parameter *Len*1. For each value of *Len*1, and let *Len*2 vary from 5 to 30, the minimum number of packets varies by only a small amount, no more than 70, for different values of *Len*2. We can also see that for each value of *Len*1, a non-reflector attack needs slightly more packets than does a reflector attack. Such experimental results are as what have been expected because any attack path from the reflector to the victim usually involves an amplified number of reflected packets which could be more than the minimum number for reconstructing the path from the victim to the reflector. Note that the number of packets from the reflectors to the victim is *nRf* times the number of packets from the attacker to the reflectors. Therefore, for a given value of *Len*1, the minimum number of packets varies only slightly for different values of *Len*2. We also found that the number of reflectors has little effect on the results as long as it is greater than ten. As for reconstruction, there is little difference between the reconstruction result of using 6000 packets and that of using 60000 packets.

Concerning the speed of attack paths reconstruction, our algorithm can reconstruct 50 distributed attack paths with lengths ranging from 20 to 30 within 5 seconds. It is obviously much faster than FMS [7]. A good portion of the reconstruction time is spent on grouping the packets. When the number of received packets becomes very large, say, more than 300,000, the reconstruction time might be somehow longer than the advanced marking scheme [9]. However, in practice, the number of marked packets is not expected to be too large; moreover, if necessary, the computation overhead on grouping the packets could be much reduced by using a sophisticated sorting algorithm and some advanced implementation techniques.

7. Conclusion

We have proposed in this paper a reflective algebraic marking scheme for tracing the ordinary DoS attacks, as well as reflector attacks. Similar to the advanced and authenticated marking scheme [3], our proposed marking scheme also makes use of an upstream routers map for speeding up attack paths reconstruction. In coping with reflector attacks, we copy marked information from incoming request packets to outgoing replied packets. The proposed marking scheme can be proved from theory and have been shown through extensive experiments that it would produce negligible false positives during attack paths reconstruction. The huge number of experiments carried out demonstrates that the marking scheme can trace effectively both the general DoS and DDoS attacks, including reflector attacks. Up to the present moment, we have not come across any published methods capable of tracing reflector attacks. When compared with other methods in the literature, our marking scheme requires relatively less packets for attack paths reconstruction. Moreover, its attack paths reconstruction could be done quite fast. We have also shown that the proposed marking scheme could be put into practice.

The most fundamental disadvantage of the proposed method is that the marked information is not authenticated. Therefore, a compromised router might tamper the information marked by its upstream routers and make the victim reconstruct wrong paths. It limits our marking scheme to reconstruct only a valid suffix of the real attack path; however a compromised router could be regarded as an attacker to a certain extent. In our future work, we shall develop a technique to authenticate the marked data so that the compromised routers could be identified. Another disadvantage is that our scheme could have difficulty to be applied to IPv6, where the IP header does not have the *Identification* field and the IP address is 128 bits. This is a inherited problem of any marking traceback technique. We believe a feasible solution could be worked out before IPv6 would become widely used.

References

[1] Drew Dean, Matt Franklin, and Adam Stubblefield, "An Algebraic Approach to IP Traceback", ACM Transactions on Information and System Security, Vol .5 No.2, May 2002, pp. 119-137.

[2] S. Savage, D.Wetherall, A. Karlin, and T. Anderson, "Practical Network Support for IP Traceback", 2000 ACM SIGCOMM Conference, Aug. 2000.

[3] D. Song and A. Perrig, "Advanced and Authenticated Marking Schemes for IP Traceback", Proc. IEEE INFOCOM 2001.

[4] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, "Numerical Recipes in FORTRAN: The Art of Scientific Computing", Cambridge University Press, 1992, pp. 83-84.

[5] Wolfgang Theilmann and Kurt Rothermel, "Dynamic distance maps of the internet", Proc. IEEE INFOCOM Conference, Vol.1, Mar 2000, pp. 275-284.

[6] I Stoica and H. Zhang, "Providing guaranteed services without per flow management", ACM SIGCOMM '99, pp. 81–94, Cambridge, MA, 1999.

[7] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Stephen T. Kent, and W. Timothy Strayer, "Hash-Based IP Traceback", Proc. ACM SIGCOMM 2001, August 2001.

[8] Thomas W. Judson. "Abstract algebra: theory and applications", Boston, MA: PWS Pub. Co., c1994, pp. 379.", Boston, MA: PWS Pub. Co., c1994, pp. 379.

[9]. Vern Paxson, "An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks", ACM Comp. Commun. Rev., vol.31, no.3, July 2001, pp. 3-14.

[10]. Lau, F., Rubin, S.H., Smith, M.H., and Trajkovic, L, "Distributed denial of service attacks" Systems, Man, and Cybernetics, 2000 IEEE International Conference on , Volume: 3 , 2000, pp. 2275 -2280 vol.3.
[11]. Chang, R.K.C., "Defending against flooding-based distributed denial-of-service attacks: a tutorial" IEEE Communications Magazine, Volume: 40 Issue: 10 , Oct 2002, pp. 42 -51.
[12]. Yamada, T., Nakamura, H., Nishimura, K., Ishizaki, T. and Ogawa, K., "A flat-time-delay transversely coupled resonator SAW filter comprising parallel connected filter tracks", Ultrasonics Symposium, 2000 IEEE , Volume: 1 , 22-25 Oct. 2000, pp. 121-124, vol.1.