International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

An IP Traceback using Packet Logging & Marking Schemes for Path Reconstruction

S. Malathi[1], B. Naresh Achari[2], S. Prathyusha[3]

[1]M.Tech Student, Dept of CSE, Shri Shiridi Sai Institute of science & Engineering, Anantapur, India

[2]Assistant Professor, Dept of CSE, Shri Shiridi Sai Institute of science & Engineering, Anantapur, India

[3]Assistant Professor, Dept of CSE, PVKK Institute of Technology, Anantapur, India

Abstract: ***The Internet has been widely applied in various fields, network security issues emerge and catch people's attention and then launch attacks. For this reason, developers have proposed a lot of trace back schemes to take out the source of these attacks. Some uses to combine packet marking with packet logging and therefore create hybrid IP trace back schemes. In packet logging no need to refresh the logged tracking information and to achieve zero false positive and false negative rates in attack-path reconstruction. In addition, we use a packet's marking field to censor attack traffic on its upstream routers. Lastly, we create and analyze our scheme, in association with other related research, in the following aspects: storage requirement, computation, and accuracy.***

Keywords: DoS attack, IP trace back, IP spoofing, packet logging, packet marking.

1. Introduction

With the fast growth of the Internet, different internet applications are developed for different kinds of users. To interrupt the service of a server, the complicated attackers may launch a distributed denial of service (DDoS) attack [1].Based on the number of packets to deny the service of a server, we can categorize DDoS attacks into flooding-based attacks and software exploit attacks. Since most edge routers do not check the origin's address of a packet, core routers have difficulties in recognizing the source of packets. The source IP address in a packet can be spoofed when an invader wants to hide him from tracing. Therefore, IP spoofing makes hosts hard to protect against a DDoS attack. For these reasons, developing a method to locate the real source of masquerade attacks has become an important issue nowadays [2].

For tracing the real source of flooding-based attack packets, we can find the upstream router through which the attack traffic passes. The victim host collects the ICMP messages to rebuild the attack path. Because earlier schemes need additional packets to trace the origin of attack packets, packet marking approaches are introduced to mark the router or path information on the triggering packets.

A packets IP header has rather limited space for marking and therefore cannot always afford to record the full route information. So, they integrate packet logging into their marking schemes by allowing a packet's marking field for the short term logged on routers. we propose a traceback scheme that marks routers' crossing point numbers and integrates packet logging with a hash table to deal with these logging and marking issues in IP traceback.

2. Related Work

Most of current single packet traceback schemes tend to log packets' information on routers. Zhang and Guan propose TOPO [9] to improve the efficiency and precision of SPIE, but TOPO still needs large storage capacity and inevitably has a false positive problem because of the bloom filter. The hybrid IP trace back schemes are introduced to mitigate the storage problem of logging-based traceback schemes. Gong and Sarac [2] propose a hybrid IP trace back scheme called Hybrid IP Traceback (HIT) [1] combining packet marking and packet logging. HIT uses packet marking to reduce the number of routers required for logging. Other researchers have proposed new schemes to further reduce the storage requirement for router logging and to decrease the number of routers required for logging, e.g., Huffman codes [5], Modulo/Reverse modulo Technique(MRT) [5] and MOdulo/REverse modulo(MORE) [7].

Even though the marking field of packet in Huffman codes, MRT, and MORE can store a path of longer length than in the fixed-length coding, the marking field may be full before the packet reaches its destination. In such a situation, they need to log the packet's information on the routers that fail to mark on the marking field. These routers then pair the packet digest with the marking field, and then they log the pair into a log table. After logging, the routers clear the marking field and repeat the marking process. When a router needs to recover the marking field of a request packet using its log table, it computes the digest of the request packet and searches the log table using exhaustive search. It could recover the marking field by the above steps. But there are the following two problems in the Huffman codes: MRT and MORE's schemes. First, after logging, if the marking field of the packet is still 0 on the adjacent downstream router, it will be identified as a logged router for the packet while tracing back. Then it will fail to find the origin. Second, since the digests in a log table might have a collision, it causes the false positive problem during the path reconstruction [5].

The storage requirement is proportional to the number of logged packets. Unfortunately, in the flooding-based attack, a huge amount of attack packets will log on the same router. Thus, it demands a high storage requirement

on the logged router. Moreover, while reconstructing a path, a logged router for a packet needs to search the digests in the log table using exhaustive search in order to find the old marking field. The exhaustive search is not efficient when the log table is big. Due to the above problems in the Huffman codes, MRT and MORE schemes, we propose a traceback scheme that marks routers' interface numbers and integrates packet logging with a hash table (RIHT). RIHT has a lower storage requirement and better precision and efficiency than Huffman codes and MRT [6]

3. IP Traceback

In this section we focus on how, the Internet has been widely applied in various fields, network security issues emerge and catch people's attention and then launch attacks [8]. For this reason, developers have proposed a lot of traceback Schemes to take out the source of these attacks. Packet marking with packet logging creates hybrid IP trace back schemes. In packet logging no need to refresh the logged tracking information and to achieve zero false positive and false negative rates in attack-path reconstruction [3].

3.1. Network Topology and Preliminaries

As the network topology shows in Fig. 1, a router can be connected to a local network or other routers, or even both. A border router receives packets from its local network. A core router receives packets from other routers. For example, R9 serves as a border router when it receives packets from Host. On the other hand, it becomes a core router when receiving packets from R8.

The assumptions of our scheme are as follows.
1) A router creates an interface table and numbers the upstream interfaces from 0 to $D(R_i)$ -1 in progress.
2) A router knows whether a packet arrives from a router or a local network.
3) Such a traceback scheme is feasible on every router.
4) The traffic route and network topology may be changed, but not often [3]

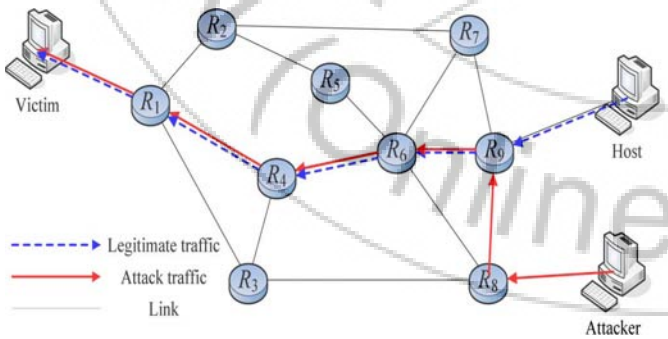

Figure 1: Network topology

3.2. Path Selection & Packet Sending

The path is said to be the way in which the selected packet or file has to be sent from the source to the destination. The Upstream interfaces of each router have to be found and it is stored in the interface table[6][2].With the help of that interface table, the preferred path between the selected source and destination can be defined. And one of the packet or file is to be selected for the conversion process. The packet is sent along the defined path from the source LAN to destination LAN. The destination LAN receives the packet and checks whether that it has been sent along the defined path or not.

3.3. Marking and Logging Scheme

When a border router receives a packet from its local network [7], it sets the packet's marking field as zero and forwards the packet to the next core router. As shown in Fig.2, when a core router receives a packet, it calculates $mark_{new}$=P. mark*(D(R_i)+1)+UI_i+1. If $mark_{new}$ is not overflow, the core router overwrites P.mark with marknew and forwards the packet to next core router [3]. If marknew is overflow, the core router must log P.mark and UIi . That is, it needs to calculate H(P.mark) first and uses a quadratic probing algorithm to search p.mark and UIi in HT . If P.mark and UIi are not found there, the core router inserts them as a pair off into the table. Then, it gets their index in the table and calculates marknew=index*(D(Ri)+1). Lastly, it rewrites P.mark with marknew andforwards the packet to the next router [6][10].

Notations

R_i	$\{R_1,R_2,\ldots,R_i,\ldots,R_x\}$, $\{R_1,R_2$ routers in the internet
$D(R_i)$	The degree of R_i
UI_r (or UI_i)	The upstream interface number of the router r (or UI_i if there is no ambiguity)
P	The received packet
H()	A hash function
M	The size of a hash table(i.e. the number of slots in a hash table)
c_1, c_2	Constants
HT	An m entries hash table
HT[index]	HT[index]: the entry of the hash table HT with the address index (HT[0] is reserved) HT[index].mark: HT[index]'s mark field HT[index].UI: HT[index]'s UI field
%	The modulo operation

Input: *P, UI_i*
begin
1. *$mark_{new}$ = P.mark * (D(R_i) + 1) + UI_i + 1*
2. ***if****$mark_{new}$ is overflow* ***then***
3. *index = h= H(P.mark)*
4. *probe = 0*
5. ***while not*** *(HT[index] is empty or HT[index] is equal to (P.mark, UI_i))*
6. *Probe++*
7. *index= (h+ c_1 * probe + c_2 * $probe^2$) % m*
8. ***endwhile***
9. ***if*** *HT[index] is empty* ***then***
10. *HT[index].mark = p.mark*
11. *HT[index].UI = UI_i*
12. ***endif***
13. *$mark_{new}$ = index * (D(R_i) + 1)*
14. ***endif***
15. *P.mark = $mark_{new}$*
16. *forward the packet to the next router*
    end

Figure 2: Marking and logging scheme

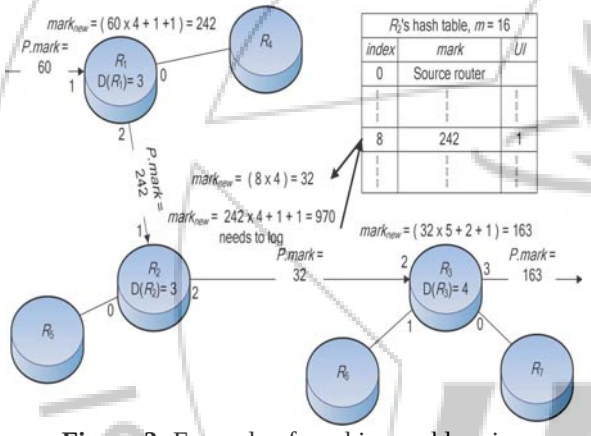

Figure 3: Example of marking and logging

Fig.3 exemplifies how to mark and log an 8-bitmarking field in our scheme. R_1 receives a packet with P.mark=60. It computes $mark_{new}$ with UI_1=1and checks whether $mark_{new}$ is overflow or not. The result shows $mark_{new}$=242, not greater than 255 and therefore not overflow, so R_1 marks the packet and forwards it to next router. R_2 receives the packet and computes $mark_{new}$.[7] Here, $mark_{new}$=970, greater than 255, and therefore overflow, so R_2 needs to log the packet. First, it takes P.mark and UI_2 as a pair and logs it into a hash table [6][7] and gets its index. Then, R_2 computes $mark_{new}$=index*(D (R2) +1). This time $mark_{new}$=32, not overflow, so R_2 forwards it to the next router. R_3 receives the packet and computes $mark_{new}$ with UI_3=2. The $mark_{new}$=163, not overflow, so R3 forwards the packet to the next router [6].

3.4. Path Reconstruction

When a victim is under attack, it sends to the upstream router a reconstruction request, which includes the attack packet's marking field, termed $mark_{req}$ here. When a router receives a reconstruction request, it tries to find the attack packet's upstream router. Depicted in Fig.4, firstly it computes UI_i=$mark_{req}$ % (D (R_i) + 1)-1. If UI_i!=-1, which means this packet came from an upstream router along the upstream interface UI_i, the requested router then restores the marking field to its premarking status. The router computes $mark_{old}$ = $mark_{req}$ / (D (R_i) + 1), so that it can get the packet's upstream router's $mark_{req}$, i.e., $mark_{old}$ here. Then replace the request's $mark_{req}$ with $mark_{old}$ and send the request to the upstream router. However, if UI_i=-1, it means either the attack packet's marking [10] field and its upstream interface number have been logged on the requested router, or the requested router itself is the source router. The requested router computes index = $mark_{req}$ / (D (R_i) + 1), so that it can decide whether the requested router is the source router or not. If index is not zero, meaning this requested router has logged[3] this packet, the router then uses index to access HT and finds $mark_{old}$= HT[index].mark and UI_i= HT[index].UI. Next, use $mark_{old}$ to replace the request's $mark_{req}$ and then send the request to the upstream router. However, if index is zero, this requested router is the source router, and the path reconstruction is done [5].

Begin
1. *UI_i = $mark_{req}$ % (D(R_i) + 1) – 1*
2. ***if*** *UI_i = -1* ***then***
3. *index = $mark_{req}$ / (D(R_i) + 1)*
4. ***if not*** *index = 0* ***then***
5. *UI_i = HT[index]. UI*
6. *$mark_{old}$= HT[index].mark*
7. *send reconstruction request with $mark_{old}$ to upstream router by UI_i*
8. ***else***
9. *this router is the nearest router to the attacker*
10. ***endif***
11. ***else***
12. *$mark_{old}$ = $mark_{req}$ / (D(R_i) + 1)*
13. *send reconstruction request with $mark_{old}$ to upstream router by UI_i*
14. ***endif***
    end

Figure 4: Path reconstruction scheme

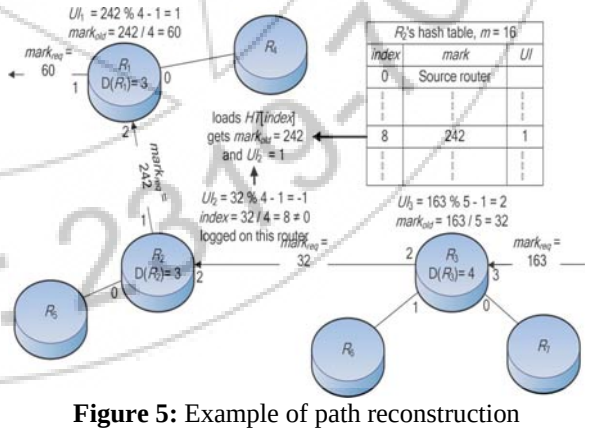

Figure 5: Example of path reconstruction

For an example in Fig.5 to explain how to reconstruct the path by using the packet's marking field. R_3 receives a request with $mark_{req}$ = 163. R_3 gets UI_3 = 163 %(4+1)-1=2! =-1, and $mark_{old}$= 163/(4+1) =32. Then it replaces $mark_{req}$ with $mark_{old}$, i.e., $mark_{req}$ in the request now becomes 32. Next, R_3 sends the modified request to R_2. When R_2 receives the request with $mark_{req}$=32, it

computes UI_2 and index, respectively, as shown in Fig.5. The results show UI_2=32%4-1 and index=32/4=8 and the packet has been previously logged in R_2's hash table [4][9][10]. R_2 uses the index to access the table and finds $mark_{old}$=242 and UI_2=1. Subsequently, R_2 uses 242 to overwrite $mark_{req}$ in the request and then sends the request to its upstream router R_1. After computation, R_1 gets UI_1=1! =-1, and $mark_{old}$=60. R_1 uses 60 to overwrite the request's $mark_{req}$ and sends the modified request to the upstream router. Note that R_4, R_5, R_6, and R_7 are not involved in this path reconstruction.

4. Analysis and Performance

4.1 Computation Analysis

In the following, I compare the computing time of logging and path reconstruction in Hybrid IP Traceback. Since Hybrid IP Traceback uses a hash table to log, inevitably have to face a hash table's collision problem. In Hybrid IP Traceback, the open addressing method is used to solve this difficulty. In this method, when a new entry has to be inserted, the slots are examined, starting with the hashed-to slot and arranged in some probe sequence, until an empty slot is found. When searching for an entry, the slots are scanned in the same sequence, until either the target record is found or a vacant slot is found. To minimize the collision problem on our scheme, I adopt the quadratic probing as the probe sequence since it requires only light calculation and is proved effective when I try to avoid clustering problem [3]. When I deal with a collision problem, I have to take into consideration a hash table's load factor, which directly affects the number of collisions. However, the calculation results of collision times may vary because it has two situations, successful search and unsuccessful search, when logging [6].Unsuccessful search means that an entry has not been logged in a hash table and as a result it is to be inserted into an empty slot. A probe is performed each time collision occurs. The estimated number of probes in unsuccessful search using open addressing is at most 1/1-∝, assuming uniform hashing. Successful search means an entry has been logged in a hash table. The expected number of probes in a successful search using open addressing is at most 1/∝ ln (1/1-∝), assuming uniform hashing[6].

4.2. Storage Requirement

This scheme maintains a hash table and an interface table on a router, while MRT and MORE maintain log tables and an interface table on a router. Since the storage requirement of an interface table is negligible, leave it out of our storage requirement study. In Hybrid IP Traceback, the size of a hash table decides how many paths can be logged on a router. For two arbitrary packets in Hybrid IP Traceback, they take the same path to a router if and only if they have the same marking filed on the router. Thus, this scheme regards the marking field of a packet as one path to a router. A hash table's load factor ∝ = l/m, where l is the number of logged paths in a hash table. As the analysis, it should maintain the load factor below 0.5. For that reason, if the number of paths which need to be logged on a router is N, then the size of the hash table on the router should be set as 2N. In addition, every entry in a hash table includes one 32-bit mark field and one 8-bit UI field; hence each entry uses 40 bits[3].

4.3. False Positive and False Negative Rates

When a router is mistaken for an attack router, it calls as "false positive" [3]. When it fails to trace back to an attacker, it call it "false negative". In MRT and MORE, the size of a log table increases with the number of logged packets, but a router's memory is limited. Thus, when those schemes are out of memory, they have to restore their log tables. The false positive or false negative problem happens when the logged data is refreshed. Unlike MRT and MORE, Hybrid IP Traceback's hash table size depends on the number of logged paths, and the table does not have to refresh. Thus, Hybrid IP Traceback has no false positive and false negative problem in this respect [3][7].

5. Conclusion

We propose a new hybrid IP traceback scheme for efficient packet logging aiming to have a permanent storage requirement in packet logging without the need to refresh the logged tracking information. Also, the proposed scheme has zero false positive and false negative rates in an attack-path reconstruction. Apart from these properties, our scheme can also organize a marking field as a packet identity to filter malicious traffic and secure against DoS/DDoS attacks. Consequently, with high accuracy, a low storage requirement, and fast computation, and also it can serve as an efficient and secure scheme for hybrid IP traceback.

Acknowledgement

I Thank Mr.B.Naresh Achari, Ms.S.Prathyusha for their comments and valuable suggestions on this manuscript.

References

[1] Ming-Hour Yang and Ming-Chien Yang, "RIHT: A Novel Hybrid IP Traceback Scheme", in IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 7, NO. 2, APRIL 2012.
[2] T. Korkmaz, C. Gong, K. Sarac, and S. G. Dykes, "Single packet IP traceback in AS-level partial deployment scenario," *Int. J. SecurityNetworks*, vol. 2, no. 1/2, pp. 95–108, 2007
[3] S. M. Bellovin, M. D. Leech, and T. Taylor, "ICMP traceback messages," *Internet Draft: Draft-Ietf-Itrace-04.Txt*, Feb. 2003.
[4] CAIDA's Skitter Project CAIDA, 2010 [Online]. Available:http://www.caida.org/tools/skitter/
[5] K. H. Choi and H. K. Dai, "A marking scheme using Huffman codes for IP traceback," in *Proc. 7th Int. Symp. Parallel Architectures, Algorithms Networks (SPAN'04)*, Hong Kong, China, May 2004, pp. 421–428.

[6] M. F. Kaashoek and D. Karger, "Koorde: A simple degree-optimal distributed hash table," in *Proc. IPTPS*, Feb. 2003, pp. 98–107.
[7] B.Al-Duwari andM. Govindarasu, "Novel hybrid schemes employing packet marking and logging for IP traceback," *IEEE Trans. Parallel Distributed Syst.*, vol. 17, no. 5, pp. 403–418, May 2006.
[8] A. Belenky and N. Ansari, "Tracing multiple attackers with deterministic packet marking (DPM)," in *Proc. IEEE PACRIM'03*, Victoria, BC, Canada, Aug. 2003, pp. 49–52.
[9] L. Zhang and Y. Guan, "TOPO: A topology-aware single packet attack traceback scheme," in *Proc. IEEE In. Conf. Security Privacy Communication Networks (SecureComm 2006)*, Baltimore, MD, Aug. 2006, pp. 1–10.
[10] D. Zhou, J. Huang, and B. Schölkopf, "Learning from labeled and unlabeled data on a directed graph," in *Proc. ICML 2005*, Aug. 2005, pp. 1036–1043.

Author Profile

S. Malathi, M. Sc, is pursuing M. Tech Computer Science &Engineering at Shri Shiridi Sai Institute of Science & Engineering, Vadiyampeta, Ananthapuramu, Andhra Pradesh, India.

B. Naresh Achari, M.Tech is working as an Assistant Professor in the Department of Computer Science & Engineering at Shri Shiridi Sai Institute of Science & Engineering, Vadiyampeta, Ananthapuramu, Andhra Pradesh, India.

S. Prathyusha, M. Tech, is working as an Assistant Professor in the Department of Computer Science & Engineering at P.V.K.K Institute of Technology, Ananthapuramu, Andhra Pradesh, India.