Record Route IP Traceback: Combating DoS Attacks and the Variants

Abdullah Yasin Nur, Mehmet Engin Tozal

University of Louisiana at Lafayette, Lafayette, LA, US

ayasinnur@louisiana.edu, metozal@louisiana.edu

Abstract

The Internet introduces a variety of vulnerabilities that put the security and privacy of computer based systems under risk. One of the most perilous threats in the Internet is the Denial of Service (DoS) attack and its variants such as Distributed DoS (DDoS). In this work we propose a novel probabilistic packet marking scheme to infer forward paths from attacker sites to a victim site and enable the victim to delegate the defense to the upstream Internet Service Providers (ISPs). We exploit the record route feature of the IP protocol to implement our probabilistic packet marking scheme. Compared to the other techniques, our approach requires less many packets to construct the paths from attacker sites toward a victim site. Our results show that a victim site can construct the forward path from an attacker site after receiving 20.23 packets on the average under DoS attacks. Moreover, we construct the forward-paths graph from 5000 attacker sites toward the victim site by receiving 11.58 packets per attacker site, on the average.

Keywords:
IP traceback, Probabilistic packet marking, Record route, Denial of service attack, Distributed denial of service attack

1. Introduction

The increasing number of cyber-attacks, acts of exploitation of computer dependent systems, is considered to be one of the most serious threats today. The economical, social and political impacts of cyber-attacks have increased tremendously along with our dependency on computer systems in the last decade. Moreover, the recent advances in Internet of Things (IoT) and Cyber-Physical Systems (CPSs) prove that this trend will only increase in the future because such systems utilize the public Internet as the main communication medium.

One of the most perilous threats in the Internet is the Denial of Service (DoS) attack and its variations. The objective of a DoS attack is to exhaust the resources of a system until the system fails to provide its usual services in a timely fashion. Typically, a DoS attack involves flooding a system by legitimate-looking traffic and making the system break down completely, work in less capacity, or fail to serve on time. However once the source of the attack is determined, it is easy to defend the system by blocking the traffic coming from the attacking site. A more severe type of the DoS attack is the Distributed Denial of Service (DDoS) attack where a large number of hosts simultaneously attack a victim site. DDoS attackers plan their attack in advance by compromising multiple hosts, scattered in the Internet, through a common vulnerability. Then, they use all compromised hosts to flood the victim site. The most recent large scale DDoS attack took place on October 21, 2016. The atack targeted the DNS infrastructure of Dyn and took some of the most popular sites offline including Netflix, Twitter, CNN, PayPal and New York Times [34].

Typically defense against DoS attacks and the variants is carried out in four phases. The first phase involves detecting any anomaly in the incoming traffic. Once a victim site detects the anomaly, the second phase is to distinguish between the rogue traffic and the legitimate traffic. The third phase involves locating the source of the rogue traffic and/or inferring the path from the source toward the victim site. In the last phase, the victim site mitigates the impact of the attack by applying its defense mechanism. Mitigating the impact of DDoS attacks by filtering the rogue traffic is difficult because it will induce too much overhead on the traffic filtering mechanism of the target system. Moreover, the attacker may forge the attack sites by using IP address spoofing. A better approach to defend a system against DDoS attacks is to delegate the defense to the Internet Service Providers on the path between the attacker and the victim sites. Therefore, it is necessary to be able to infer the forward paths from attack sites to the victim site. The content of this work falls in the third phase of DoS defense: locating the source of the rogue traffic and/or inferring the path from the source toward the victim site.

Inferring the path between two hosts from the destination site in the Internet is called IP traceback. The IP traceback problem is a challenging one because it is not directly supported by the IP protocol. Reverse traceroute [24] takes the advantage of routers that copy the IP header in their ICMP response messages to reveal a forward path. It requires densely distributed vantage points as well as IP spoofing support to reveal longer forward paths in the Internet. Although the technique does not require any modification in packet forwarding, it suffers from rate limiting practices, restricted scalability and limited router support [25] [26] [27].

Other solutions that require modifications in packet forwarding scheme are based on packet marking. Packet marking is blemishing the packets with routers' IP address information while the packets traverse the routers from the source toward the destination. There are two different packet marking schemes; Probabilistic Packet Marking (PPM) [4] and Deterministic Packet Marking (DPM) [12]. In PPM the routers on a path probabilistically infuse their IP address information into the packet. On the other hand, in DPM only the ingress routers on a path mark the packets passing through them with their IP address information. In this work we propose a novel packet marking scheme based on probabilistic packet marking.

We modify the record route feature of the IPv4 protocol [13], yet our scheme can be extended to IPv6 as well [14]. Typically, each router forwarding a packet in the Internet checks the record route option of the packet. If the record route option is enabled, the router inserts one of its IP addresses (preferably IP address belonging to the outgoing interface) into the options field of the packet header. Due to the restrictions regarding the size of the IP packet header [13], at most nine IP addresses can be stored in the options field. Hence, if the options field of the IP packet header is full, the router skips inserting its IP address.

In our probabilistic scheme, a router always inserts its IP address as long as there is room in the IP options field. On the other hand, if the IP options field is full, a router probabilistically restarts record routing by clearing the options field of the IP packet header. The destination site on the other hand, gradually joins the recorded IP addresses in the options field to construct a forward-paths graph from all sources toward the destination. Basically, the destination site starts with an empty forward-paths graph and fills the graph with subpaths that are reported via record route. After receiving a packet, the destination site aligns the sequence of the IP addresses in the record route with respect to the current snapshot of the graph and implants the new route in the graph. The final graph can be used to reduce the impact of DoS/DDoS attacks in real time by delegating the defense to ISPs. Our approach focuses on building a forward-paths topology to fortify DoS/DDoS defense systems rather than discriminating rogue traffic from legitimate traffic.

We conducted several experiments using a real world dataset to demonstrate the efficiency of our approach under DoS and DDoS attacks. Our results show that a victim site can construct the forward path from an attacker site after receiving 20.23 packets on the average under DoS attacks. In DDoS attacks, the average number of packets needed to construct a forward-paths graph decreases as the number of the attacker sites increases due to the overlapping paths toward the victim site. To illustrate, it is enough to receive 11.58 packets per attack site to construct a forward-paths graph for 5000 attackers. In addition, we compared our results to other probabilistic packet marking schemes: Fast Internet Traceback (FIT) [7], Authenticated Marking Scheme (AMS) [5] and Fragment Marking Scheme (FMS) [4]. Our results show that FIT needs between 244 and 858 packets, AMS needs between 400 and 1400 packets and FMS needs between 1000 and 4100 packets on the average to construct forward paths of varying hop distances. On the other hand, our approach requires 1 to 229 packets on the average for the same path traces. The overall averages are 20.23, 457.18, 839.83 and 2253.38 for our approach, FIT, AMS and FMS respectively, regardless of the hop distance. Compared to the other probabilistic packet marking schemes our approach (i) requires less many packets to construct a forward path from an attack site toward the victim site; (ii) related to the previous item, it takes less time to construct the forward path; (iii) it does not introduce any false positive or negative paths; (iv) it is resistant to IP spoofing, i.e., we detect the true forward path even if the attacker spoofs its IP address; finally, (v) it needs even less many packets to construct forward paths as the number of attackers increases. Reducing the required number of packets to construct forward paths is especially important as adversaries have already started to compromise IoT devices to conduct large scale DDoS attacks [32]. Larger pools of compromised devices allow the adversaries to fabricate rogue packets sporadically at random to reduce the number of packets per attack site and make attack detection and defense more difficult.

Our approach complies with the current IP protocol and does not require any changes in the protocol itself. Similar to other packet marking schemes, the proposed approach in this study requires support from router vendors and Internet Service Providers. The experimental results show that our approach can be very effective against DoS and DDoS attacks with a small overhead on the routers.

A preliminary work of this study appeared in IEEE SmartComp as a work-in-progress paper [1]. In this study, we present a theoretical model for our scheme; introduce an alternative rewrite-skip algorithm; develop an efficient forward-paths graph construction algorithm; extend our experiments to DDoS attacks; and compare our approach with FIT [7], AMS [5] and FMS [4].

The rest of the paper is organized as follows. In Section II, we present the related work. We explain the details of our approach in Section III. In Section IV, we present our forward-paths graph construction algorithm. Section V demonstrates our experimental results. We discuss the advantages and limitations of our approach in Section VI. Finally, in Section VII we conclude the paper.

2. Related Work

Many approaches have been suggested to defend systems against DoS attacks and the variants [20], [21], [22], [23]. Mirkovic and Reiher classified these attacks and their defense mechanism [16]. One of the suggested solutions named *IP traceback* is about finding the paths between the attackers and the victim site. The IP traceback problem has been extensively studied in the last two decades [4], [5], [7], [11], [15], [19].

The earliest work can be credited to Burch and Cheswick [3]. Their approach is based on mapping the paths from the victim site towards the attacker sites and flooding the routers on the path to detect the source(s) of an attack. This technique assumes the paths in the internet are symmetric and introduces extra load to determine attacking site(s).

Savage et al. proposed fragment marking scheme (FMS) in one of the earliest works [4]. They used 16-bit IP identification field to probabilistically mark the router information. The victim site uses the router information to construct forward paths.

Song et al. [5] proved that FMS approach requires many packets to construct the forward paths and introduces many false positives for DDoS attack. They improved the FMS approach to reduce the number of required packets for constructing the forward paths as well as to decrease the false positive rate.

Dean et al. [6] proposed an algebraic approach to solve the IP traceback problem. Their algorithm encodes router's IP address as a polynomial in 16 bit IP Identification field. In a later study, Yaar et al. [7] proved that this approach is not efficient for large number of attackers. Although their approach is similar to the latter, they use more space for encoding to decrease the number of false positives and reduce the number of the required packets.

Belenky and Ansari proposed a deterministic packet marking scheme [12] and they improved their work in a later study [18]. In their method, the ingress routers on a path mark every packet passing through them with their IP address information.

Yaar et al. [8] proposed a path identification mechanism based on deterministic packet marking idea. They mark each packet with an identifier along a path. Packets traveling along the same path have the same identifier. They use the identifiers to filter packets taking different paths.

Foroushani et al. [17] divided the problem into three main components: detection, traceback, and traffic control. The detection component involves identifying unusual, suspicious activities in the incoming traffic. Then, they use deterministic flow marking techniques to find attackers. Lastly, they send messages asking for traffic adjustment to the edge routers of the attacking networks.

Katz-Bassett et al. proposed reverse traceroute [24] to infer a forward path from a destination to a source at the source host. Their approach relies on PlanetLab [33] vantage points and IP spoofing support to reveal longer forward paths in the Internet. Although the technique does not require any modification in packet forwarding, it suffers from rate limiting practices, restricted scalability and limited router support [25] [26] [27]. More importantly, reverse traceroute fails to traceback the source of an attack when IP spoofing is used to hide the location of the attack.

In this paper, we propose a novel probabilistic packet marking scheme to infer forward paths from attacker sites to a victim site. We modify the record route feature of the IP protocol. Our results show that the victim site can construct a single forward path by 20.23 packets on the average.

3. Methodology

In this study, we exploited the record route feature of the IP protocol and the probabilistic packet marking (PPM) technique. IP address protocol provides record route option to record the route of an internet packet. An IP packet with enabled record route option solicits each router that handles the packet to append its IP address into the options field. Hereby, the destination host can get the list of the routers appeared on the forward path. Note that the maximum number of IP addresses that could be stored in the IP options field is nine [13]. Therefore, the destination host receives the IP addresses of the first nine hops even if the path between the two hosts is longer than nine.

Probabilistic packet marking schemes involve the routers that handle an IP packet to embed information into the packet. Typically the embedded information reflects the identity of the router or the link that the packet passes through.

Our scheme requires a router to probabilistically modify the options field of an IP packet. Similar to the IP record route, a router appends its IP address (preferably IP address belonging to the outgoing interface) into the options field of a packet when there is room. Different from the IP record route, a router rewrites the options field of a packet with probability p when the options field is full. Equivalently, the router skips rewriting the options field of a full packet with probability $1 - p$. A router rewriting a full packet, swaps the first IP address in the options field by the last IP address and erases the rest of the options field. Then, It appends its own IP address in the options field.

Figure 1 shows an example path consisting of 17 routers between an attacker and a victim host. In accordance with our rewriting scheme, the first nine routers append their IP addresses in the record route options field of the packet, $[R_1, R_2, \ldots, R_8, R_9]$. The tenth router on the path rewrites the packet with probability p or skips rewriting with probability $1 - p$. In case it rewrites the packet, the eleventh router receives $[R_9, R_{10}]$ in the record route options field and appends its IP address with probability 1 since there is room in the field. As a result, the next router receives $[R_9, R_{10}, R_{11}]$ in the options field. On the other hand, if the tenth router does not rewrite the packet the eleventh router receives $[R_1, R_2, \ldots, R_8, R_9]$ in the record route options field. Again, the eleventh router rewrites the packet with probability p or skips rewriting with probability $1 - p$. If it rewrites then the next router receives $[R_9, R_{11}]$ in the record route options field. Otherwise, the next router receives $[R_1, R_2, \ldots, R_8, R_9]$. This procedure repeats itself until the packet reaches to the victim site.

The tree structure in Figure 1 demonstrates all possible record route options field for a path between the attacker and the victim sites. Each level in the tree represents packets that might be received by the corresponding router. The arrow(s) between the levels of the tree represent the probabilistic decisions that might be made by a router before forwarding the packet to the next router. Note that each arrow is labeled by its decision probability except the for sure cases where the probability is one. Finally, the leaves of the tree represent all possible record route options fields that the victim site may receive.

The leaves marked by thick lines in the tree represent the *essential* record route (essentialRR) IP lists that the victim needs to receive to rebuild the forward path correctly. The record route IP lists other than the *essentialRRs* (non-essentialRRs) miss one or more intermediate routers on the forward path. On the other hand, *essentialRRs* hold a complete sequence of routers on the forward path with no missing intermediate router. Therefore, it is necessary for the victim site to receive all *essentialRRs* and concatenate them in the right order to build the forward path. It is important to note that the last IP address in an *essentialRR* is the first IP address in the next *essentialRR*.

Additionally, we use two control bits in the IP packet options field to stamp packets with a packet type. Control bits '00', '01' and '10' denote *essentialRR* packets while '11' represents *non-*

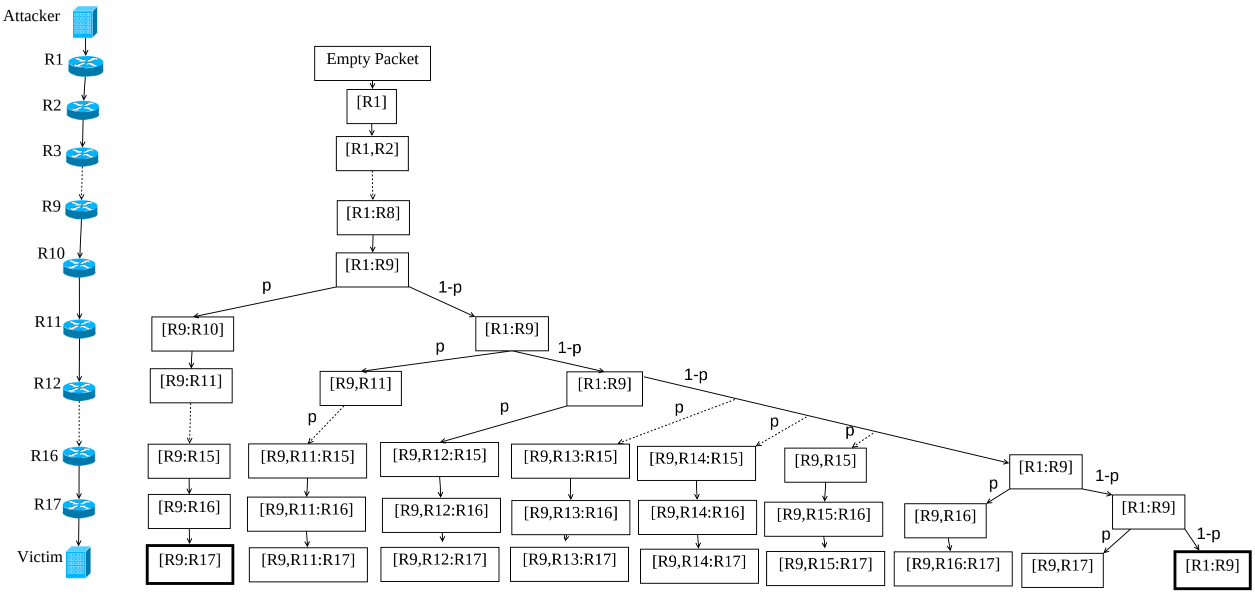

Figure 1: All possible record route options field for an example path between the attacker and the victim sites

essentialRR packets. The control bits are initially set to '00' when a packet is created and they are subject to change by the routers handling the full packets.

- '00' denotes the initial control bits. Note that in case the hop distance between the attacker and the victim site is less than or equal to nine, the victim site only gets an *essentialRR* packet with '00' control bits because there is no rewrite or skip-rewrite operation.
- '10' stands for the last *essentialRR* packet where the hop distance is greater than nine.
- '01' denotes any *essentialRR* packet except the last one where the hop distance is greater than nine.
- '11' stands for any *non-essentialRR* packet.

Algorithm 1 presents the rewrite and skip-rewrite pseudocode. The algorithm expects the record route options field (*RROptions*) of an IP packet consisting of an *RRIPAddr* sub-field accommodating a vector of IP addresses and a *ControlBits* sub-field holding two control bits that represent the packet type. Also it expects the IP address of the current router handling the packet as input. Finally, it returns an updated record route options field (*RROptions*). If the router probabilistically decides to rewrite, the algorithm executes lines 1 to 12. In case the router skips rewriting, the algorithm executes lines 13 to 21.

At line 2, the algorithm stores the last IP address in the record route options vector into a temporary variable. It erases the elements of the vector at line 3. At line 4 and 5, it appends the IP address stored in the temporary variable and the IP address of the current router. Finally, it updates the control bits between the lines 6 and 12.

If the hop distance between the attacker and the victim sites is less than or equal to nine, any received packet from the attacker is considered to be an *essentialRR*. In case not, a packet is an *essentialRR* packet only if all routers skip rewriting when the first router receiving a full packet skips rewriting. Otherwise, the packet is considered to be a *non-essentialRR* packet. Line 7 is executed when the 10th router decides to rewrite to denote that the hop distance is larger than nine. Line 9 is executed by a rewriting router which is not the very first one receiving a full packet to denote that it is a *non-essentialRR* packet. Lastly, line 11 is executed by a rewriting router to denote that the packet is either *non-essentialRR* or the last *essentialRR*.

Algorithm 1 Rewrite / Skip-Rewrite Algorithm

```
Input: $RROptions$                          ▷ IP record route options field
Input: $IP$                                 ▷ Current router IP address
Output: $RROptions$                 ▷ updated record route options field
 1: if Router decides to rewrite then
 2:     $temp = RRIPAddr.get(9)$
 3:     $RRIPAddr.erase()$                         ▷ erase all IP addresses
 4:     $RRIPAddr.append(temp)$
 5:     $RRIPAddr.append(IP)$
 6:     if $ControlBits$ equal to $'00'$ then
 7:         update to $'10'$
 8:     else if $ControlBits$ equal to $'01'$ then
 9:         update to $'11'$
10:     else                                      ▷ '10' and '11' case
11:         do not update bits
12:     end if
13: else                               ▷ router decides to skip rewriting
14:     if $ControlBits$ equal to $'00'$ then
15:         update to $'01'$
16:     else if $ControlBits$ equal to $'10'$ then
17:         update to $'01'$
18:     else                                      ▷ '01' and '11' case
19:         do not update bits
20:     end if
21: end if
```

Similarly, line 15 is executed when the 10th router decides to skip-rewrite to denote that the hop distance is larger than nine. Line 17 is executed by a skip-rewrite router to denote that the packet is an intermediate *essentialRR*. Lastly, line 19 is executed by a skip-rewrite router to denote that the packet is either a *non-essentialRR* or the last *essentialRR*. Note that, once the *ControlBits* is set to '11', it remains as '11' to denote that the packet is not an *essentialRR*, i.e., the *RRIPAddr* sub-field misses one or more routers on the path.

3.1. Probabilistic Model

Let `essentialRRV` be an ordered list of *essentialRR* packets that are needed to build a forward path from the attacker toward the destination. Given that the hop distance from the attacker to the victim site is h, where $h > 1$, the total number of the required *essentialRRs* is $\lceil \frac{h-1}{8} \rceil$. By definition the total number of the required *essentialRRs* is one when $h = 1$. Let k, $1 \leq k \leq \lceil \frac{h-1}{8} \rceil$, be the position of an *essentialRR* in the list such that `essentialRRV[k]` refers to the k^{th} *essentialRR* packet. Note that all *essentialRRs* except the last one hold nine IP addresses. Assuming that the hop distance from the attacker host to the victim site is h, the last *essentialRR* always contains $h - 8(k-1)$ IP addresses.

Moreover, let i, $1 \leq i \leq 9$, be the index of an IP address located in an *essentialRR*. That is, `essentialRRV[k][i]` denotes the IP address located in the k^{th} *essentialRR* at position i. The hop distance of the IP address at the `essentialRRV[k][i]` can be calculated by $8(k-1) + i$.

To illustrate, let the hop distance between two hosts be 20. The required number of *essentialRRs* is $\lceil \frac{20-1}{8} \rceil = 3$. The first two *essentialRRs* contain the IP addresses located at hops $[1:9]$ and $[9:17]$, respectively. The third *essentialRR* contains four IP addresses, $20 - 8(3-1) = 4$, located at hops $[17:20]$.

In order to calculate the average number of the packets to build a forward path, we need to compute the probability to receive each *essentialRR*. In the following we detail the probability of receiving *essentialRRs*.

Remember that as long as the record route options (*RROptions*) field of an IP packet is not full, each router appends its IP address in the *RROptions*. In case the *RROptions* field is full, a router rewrites it with probability p and skips rewriting with probability $1 - p$ (Algorithm 1). Note that all *essentialRRs* strictly hold nine IP addresses except the last one. The last *essentialRR* contains nine IP addresses if the number of the hops between the two hosts is equivalent to $h \equiv 8x + 1$ where $x \in \mathbb{Z}^+$. Otherwise, the last *essentialRR* contains $h - 8(k-1)$ IP addresses.

In order for the destination to receive the k^{th} *essentialRR*, there must be $k - 1$ rewrites *immediately after the* RROptions *field of a packet gets full*. Since the last rewrite preserves two IP addresses in the *RROptions*, the following routers append their IP addresses until the *RROptions* is full, once again. Once the *RROptions* of the k^{th} *essentialRR* packet is full, the remaining routers (if any) must skip rewriting.

Hence, the probability of receiving the k^{th} *essentialRR*, $P\{k\}$, is

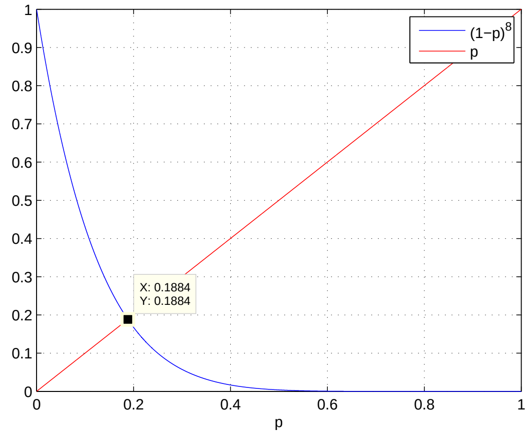

Figure 2: Best value for p

$$P\{k\} = \begin{cases} p^{k-1} \times 1^{7k+2} \times (1-p)^{h-(8k+1)}; \, if \; k < \lceil \frac{h-1}{8} \rceil \\ p^{k-1} \times 1^{h-k+1}; \, otherwise \end{cases} \tag{1}$$

where $\lceil \frac{h-1}{8} \rceil$ denotes the rank of the last *essentialRR*. In Equation 1, the first $k - 1$ routers receiving a full *RROptions* rewrite with probability p and the following routers receiving a full *RROptions* skip rewriting with probability $1 - p$. Moreover, the terms 1^{7k+2} and 1^{h-k+1} are syntactic sugars representing the routers that append their IP addresses with probability 1 when there is room in the *RROptions*. Note that in case k refers to the last *essentialRR* the second piece of the function in Equation 1 computes the relevant probability because there is not any routers skipping the rewrite operation after the last rewrite.

$$P\{essentialRR\} = \sum_{k=1}^{\lceil \frac{h-1}{8} \rceil} P\{k\} \tag{2}$$

Equation 2 denotes the probability of receiving an *essentialRR*. Maximizing Equation 2 in terms of p allows us to increase the probability of receiving an *essentialRR*. On the other hand, the resulting probability scheme may favor some of the *essentialRRs* compared to the others which may cause disfavored *essentialRRs* appear rarely. A better approach would be assigning a uniform-like probability to each *essentialRR* as shown in Equation 3

$$\begin{aligned} P\{k\} &= P\{k+1\} \\ p^{k-1} \times (1-p)^{h-(8k+1)} &= p^k \times (1-p)^{h-(8(k+1)+1)} \\ (1-p)^8 &= p \\ &\Rightarrow p = 0.1884 \end{aligned} \tag{3}$$

where $1 \leq k < \lceil \frac{h-1}{8} \rceil$. Figure 2 shows the optimal value of p with respect to Equation 3. Our experimental results show that the optimal $p = 0.1884$ works well for all *essentialRRs* including the last *essentialRR*.

3.2. Expected Number of Packets

In this section we study the expected number of packets needed to construct a single forward path at the victim site. Our approach can be considered as a modification of the classical Coupon Collector's Problem [9]. The Coupon Collector's Problem involves the usual setup for a "collect all coupons to win" contest. Assume that there are k different coupons that one has to collect to win a prize. Each coupon has the same probability of being drawn and a person draws a new coupon each time with replacement. The Coupon Collector's Problem examines the average number of the required draws to collect each type of coupon at least once.

In our problem each *essentialRR* corresponds to a coupon. The destination site needs at least one instance of each *essentialRR* to construct a single forward path. Unlike the classical problem, the destination site receives *non-essentialRR*s as well. Moreover, the probability of receiving a *non-essentialRR* is not equal to the probability of receiving an *essentialRR*.

Similar to the previous section, let h be the number of hops between a source and a destination. Then, the total number of distinct *essentialRR*s is $m = \lceil \frac{h-1}{8} \rceil$. Let $P\{k\}$ given in Equation 1 be the probability of receiving the k^{th} *essentialRR*. Regardless of the order of *essentialRR*s, assume that $i-1$ *essentialRR*s have already been collected at least once. Then, the probability of receiving a new *essentialRR* that was not collected before is $p_i = [m-(i-1)] \times P\{k\}$. To put in other words, p_i is the probability of receiving the i^{th} unobserved *essentialRR* after observing $i-1$ *essentialRR*s regardless of the order.

Let T_i be a random variable denoting the number of the packets to be received in order to get the i^{th} new *essentialRR*, given that $i-1$ *essentialRR*s have already been collected. The p.m.f of T_i is $P\{T_i = t\} = p_i \times (1-p_i)^{t-1}$. The expected value of $E[T_i]$ (Equation 4) gives us the average number of packets to observe a new *essentialRR*, given that $i-1$ *essentialRR*s have already been collected.

$$
\begin{aligned}
E[T_i] &= \sum_{t=1}^{\infty} t \times p_i \times (1-p_i)^{t-1} \\
&= p_i \times \sum_{t=1}^{\infty} t \times (1-p_i)^{t-1} \\
&= p_i \times \frac{\partial}{\partial(1-p_i)} [\sum_{t=1}^{\infty} (1-p_i)^t] \\
&= p_i \times \frac{\partial}{\partial(1-p_i)} [\frac{1-p_i}{p_i}] \\
&= p_i \times \frac{1}{p_i^2} \\
&= \frac{1}{p_i}
\end{aligned} \qquad (4)
$$

where the final step is calculated by differentiating and integrating the right hand side of Equation 4 with respect to $1-p_i$.

Finally, $T = \sum_{i=1}^{m} T_i$ is the total number of received packets to collect all *essentialRR*s at least once. That is, $E[T]$ (Equation 5) is the total number of packets required to construct a complete forward path using all *essentialRR*s.

$$
\begin{aligned}
E[T] &= \sum_{i=1}^{k} E[T_i] \\
&= \sum_{i=1}^{k} \frac{1}{p_i}
\end{aligned} \qquad (5)
$$

Note that $E[T]$ in Equation 5 gives the expected value for $h \equiv 8x+1$ which also serves as an upper bound for the cases that do not satisfy the equivalence. To illustrate, it requires 1.00, 7.96 and 51.68 packets on the average for hop distances 9, 17 and 25, respectively. At most 7.96 packets are needed to construct a forward path between two hosts if they are 9 to 17 hops apart.

4. Forward-Paths Graph Construction

The victim site gradually joins the recorded IP addresses in the options fields of the received packets to construct a forward-paths graph from all sources including the attacker sites. The forward-paths graphs is a directed graph representing multiple paths from sources toward the victim site. The victim site starts with an empty forward-paths graph and fills the graph with sub-paths that are reported via record route options field. After receiving a packet, the victim site aligns the sequence of the recorded IP addresses with respect to the current snapshot of the graph and implants the new route in the graph. In order to form a complete forward path from a single source, the victim site needs to get at least one instance of each *essentialRR* on the path. Remember that, the victim site uses the *ControlBits* sub-field to determine the type of a received packet (Section 3). Moreover, *essentialRRs* hold a complete sequence of routers on the forward path with no missing intermediate routers. On the other hand, *non-essentialRRs* miss one or more intermediate routers on the forward path. The missing routers can only appear between the first and the second reported IP addresses in a *non-essentialRR* packet because a rewrite operation after one or more skipping routers allow room for the following routers to append their IP addresses. To put in other words, all IP addresses after the first IP address in a *non-essentialRR* packet belong to consecutive routers in our scheme. Our forward-paths graph construction algorithm utilizes those IP addresses in *non-essentialRRs* in addition to the ones in *essentialRRs* to minimize the required number of the packets to build a forward-paths graph.

Algorithm 2 presents the forward-paths graph construction pseudocode. The algorithm expects the record route options (*RROptions*) field of a received packet, as well as the current snapshot of the forward-paths graph. It updates the input graph by inserting the links of the path in the *RROptions* field. At lines 1-5, the algorithm checks if the received packet holds an *essentialRR* or not and sets the alignment index that marks the beginning of the consecutive IP addresses. The for loop at line 6 processes each pair of consecutive IP addresses in the *RROptions* field. Specifically, lines 7 and 8 define the variables holding two consecutive IP addresses. Lines 9 and 12 check if the

Algorithm 2 Forward-Paths Graph Construction

```
Input: $G(V, E)$                              ▷ current graph
Input: $RROptions$   ▷ IP record route options field of received
    packet
Output: $G(V, E)$                             ▷ updated graph
 1: if $ControlBits\ is\ '11'$ then
 2:     $startIndex = 1$                     ▷ non-essentialRR
 3: else                                         ▷ essentialRR
 4:     $startIndex = 0$
 5: end if
 6: for $i = startIndex\ to\ RRIPAddr.length - 1$ do
 7:     $IP_i = RRIPAddr.get(i)$
 8:     $IP_{i+1} = RRIPAddr.get(i + 1)$
 9:     if $IP_i$ not in the graph then
10:         create new $IP_i$ node
11:     end if
12:     if $IP_{i+1}$ not in the graph then
13:         create new $IP_{i+1}$ node
14:     end if
15:     if no link between $IP_i$ and $IP_{i+1}$ in the graph then
16:         insert link from $IP_i$ to $IP_{i+1}$
17:     end if
18: end for
```

related IP addresses are already in the graph and adds them in case not. Finally, line 15 adds the link between those IP addresses if it is not already in the graph.

The victim site keeps the forward-paths as a graph in its memory and updates it as the packets arrive. In general, graphs are either represented as adjacency matrices or adjacency lists. In our implementation, we represented the forward-paths graph as an adjacency list because the the IP-level graph representation of the Internet is sparse. Therefore, our memory complexity is O($|V|$ + $|E|$) where V represents the IP addresses and E corresponds to the links between the routers of the two consecutive IP addresses appearing on a path trace. Obviously, the practical memory complexity depends on the implementation as well as the implementation language. The time complexity of the algorithm is O($|V|$ + $|E|$) because for each IP address the algorithm locates the successor among the neighbours of the IP address.

5. Experimental results

In this section we empirically demonstrate the efficiency of our algorithm using a real world dataset. We used the iPlane interface-level atlas dataset [10] consisting of more than 15 million (15,865,206) path traces collected from more than 500 vantage points. The dataset consists of $468,179$ different IP addresses and $2,067,025$ links between them. iPlane collects this dataset by sending traceroute probes to all routable IP prefix clusters [29]. Note that we only included the loop-free path traces that reach to their specified destinations. The minimum and maximum hop lengths in our dataset are 1 and 31, respectively. The average hop length is 16.23 and the hop length distribution is symmetric-like as shown in Figure 3. Moreover, we assume that the routers on a path between an attacker and a victim site are not compromised. In case the routers along the attacking paths are compromised our technique will fail like any other probabilistic packet marking scheme.

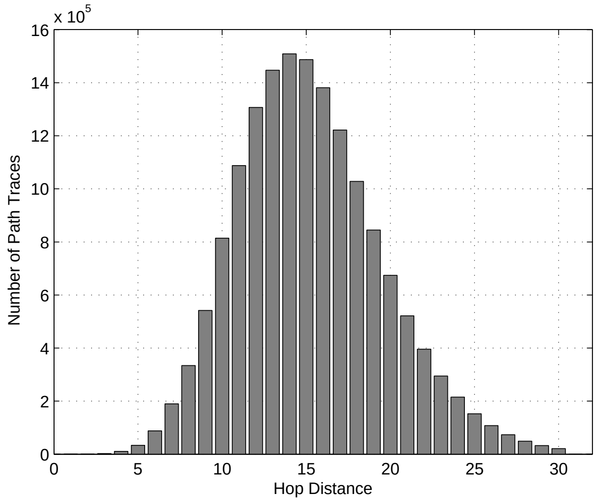

Figure 3: iPlane Hop distance distribution

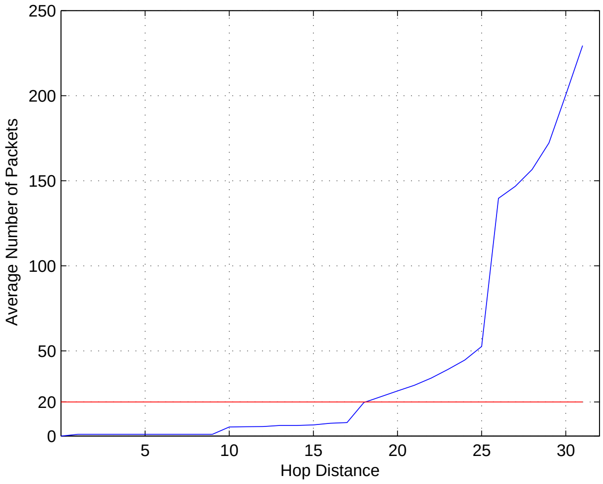

Figure 4: Average number of packets needed to construct a path from an attacker toward a victim site with respect to the hop distance. The red line shows the overall average at 20.23 packets

We first conducted two experiments for DoS and DDoS. In the DoS case, we randomly picked a single attacker and a victim IP addresses. Next, we ran our Matlab procedure which emulates the approach presented in Section 3. We gradually constructed the forward-paths graph from the attacker IP address toward the victim IP address.

Our results show that a victim site can construct the forward path from a single attacker site after receiving 20.23 packets on the average. However, the number of packets needed to build a forward path depends on the hop distance between the attacker and the victim site. Figure 4 shows the average number of packets needed to construct a forward path with respect to the hop distances between the attacker and victim sites. Obviously, the number of packets needed to build a forward path increases (non-linearly) as the distance increases. Particularly, it reaches to 229 packets at hop distance 31. Moreover, the figure

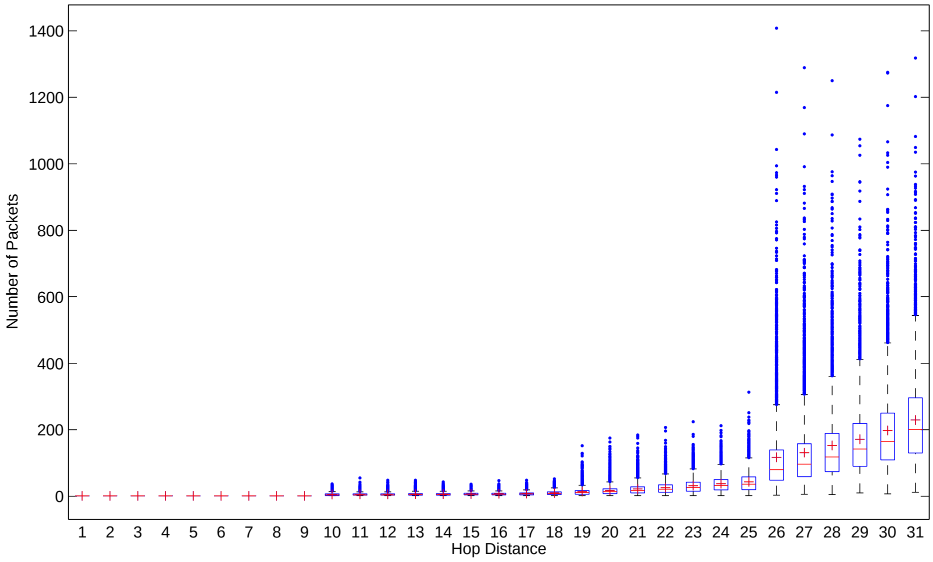

Figure 5: Empirical distribution of the number of the required packets per hop distance to construct a forward path

demonstrates jumps at hop distances 9, 17 and 25 according to $h \equiv 8x+1$ where $x \in \mathbb{Z}^+$ (Section 3). Because the number of the required *essentialRRs* increases after the boundaries 9, 17 and 25, meaning that more packets needed to construct a forward path. In theory, the required number of packets to construct a path grows non-linearly as the hop distance increases. However, this theoretical disadvantage does not affect our scheme in practice because the diameter of the Internet is reported to be around 30 [28]. This figure complies with our extensive iPlane dataset which reveals maximum hop distance as 31 out of more than 15 million path traces. Considering that the diameter of the Internet is 31 in our iPlane snapshot, the non-linear increase in the average required number of packets is bounded by 229 in practice.

To provide the reader with more insight, Figure 5 shows box plots of the required number of packets per hop distance. In this experiment, we randomly chose 5000 path traces per each hop distance. Next, we computed the distribution of the number of the required packets with respect to the hop distances. In the figure, the box plots show the first, second (median) and the third quartiles. For each hop distance, the average number of packets is shown by a '+' sign. The outlying cases are shown by dots. In compliance with Figure 4, the average number of packets increases as the hop distance increases. Figure 5 shows that the variance also increases as the hop distance increases beyond 26. Obviously, this introduces outlying cases where the required number of packets changes from 275 to 1408 for hop distances 26 thru 31. However, the ECDF plots in Figure 6 show that in 95% of the cases the number of the required packets change between 350 and 494. Moreover, reading Figure 5 along with Figure 3 shows that not only the probabilities of having the outliers are low but also the number of long hop distances are small in the Internet. That is, in Figure 3 the paths of length 26 or more only corresponds to the 2.82% cases. As a result the joint probability of having a long path and an outlier case becomes very small.

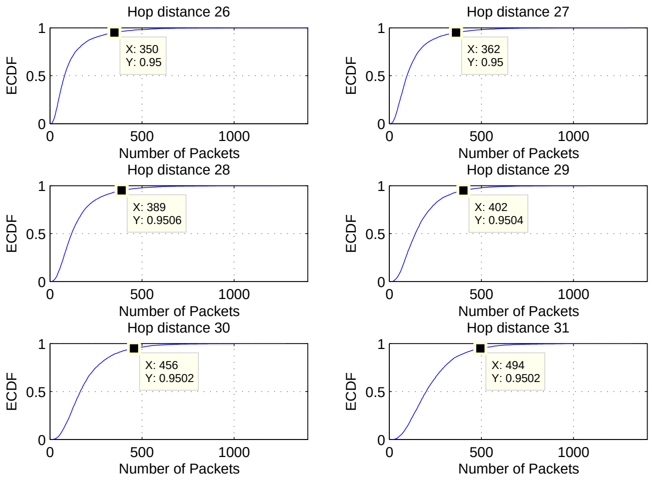

Figure 6: ECDF for hop distances 26 to 31 out of 5000 samples

Table 1 shows theoretical (Section 3.2) and experimental average number of packets needed to build a forward path between an attacker and a victim site. The table shows average number of the required packets for boundary hop distances 9, 17, 25 and 33. For any other hop distance between any of the two boundaries the larger boundary gives the upper limit of the average number of the required packets. The theoretical results are slightly different from the experimental results due to the randomness factor in our experiments. Lastly, there is no experimental figure for hop distance 33 in the table because the maximum hop distance in our real world dataset is 31.

Probabilistic packet marking approaches in the literature use the IP ID field of IP packets which is two bytes. Because those approaches either fragment IP addresses or their hashes or encode consecutive IP addresses, they may introduce false positive or negative paths. On the other hand, our approach

Table 1: Comparing theoretical and experimental average number of packets w.r.t. boundary hop distances

Hop Distance	Average number of packets (Theoretical)	Average number of packets (Experimental)
9	1	1
17	7.9640	7.9260
25	51.6798	51.6030
33	311.8013	-

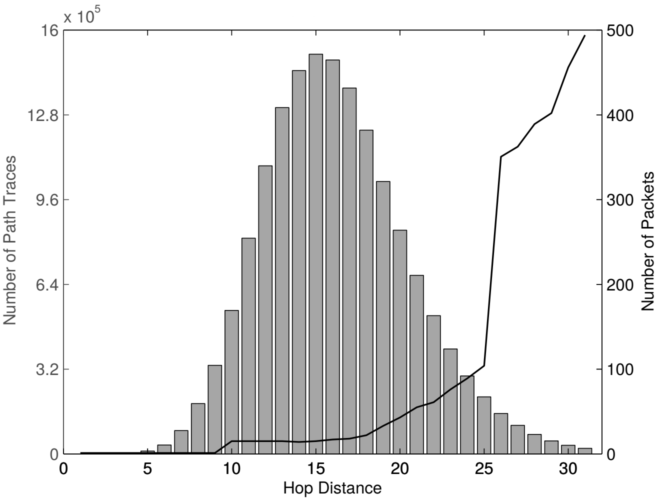

Figure 7: Complete path construction at 95th percentile

stitches partial paths together such that each partial path is correct. Moreover, the forward path construction procedure given in Algorithm 2 stitches the partial paths according to their beginning and ending IP addresses. Therefore, our approach does not produce any false positives or negatives. However, at any time the forward-paths graph may have incomplete paths due to the lack of the required packets. As a result, we analyse our approach in terms of partial/complete paths rather than false positives/negatives. Figure 7 shows the number of the packets required to fully construct a forward path from an attacker toward a victim at 95% of the time. Since the number of the required packets depends on the hop distance, we superimpose the hop distance frequency along with 95 percentile line. In the figure, the required number of packets to construct 95 percentile forward paths increases as the hop distance increases. Particularly, it jumps from 104 to 350 at hop distance 26. However, as shown in the same figure the number of path traces having hop distance 26 or more corresponds to only 2.82% of all path traces.

In our next experiment, we emulated DDoS attack for 100, 1000, 2000, 3000, 4000 and 5000 randomly chosen attackers. Instead of assuming that each attacker conducts an independent DoS attack, we gradually augmented the forward-paths graph after receiving a packet regardless of its source. In this way, we built the DDoS forward-paths graph by using less many packets on the average because the forward paths from attacker sites tend to merge towards the victim site. Figure 8 illustrates a DDoS attack conducted from 14 attacker sites toward a victim site. In the figure, the paths taken by the DDoS packets merge at various routers. The link colors between the routers reflect the merging traffic toward the victim site using a color gradient from light gray to black.

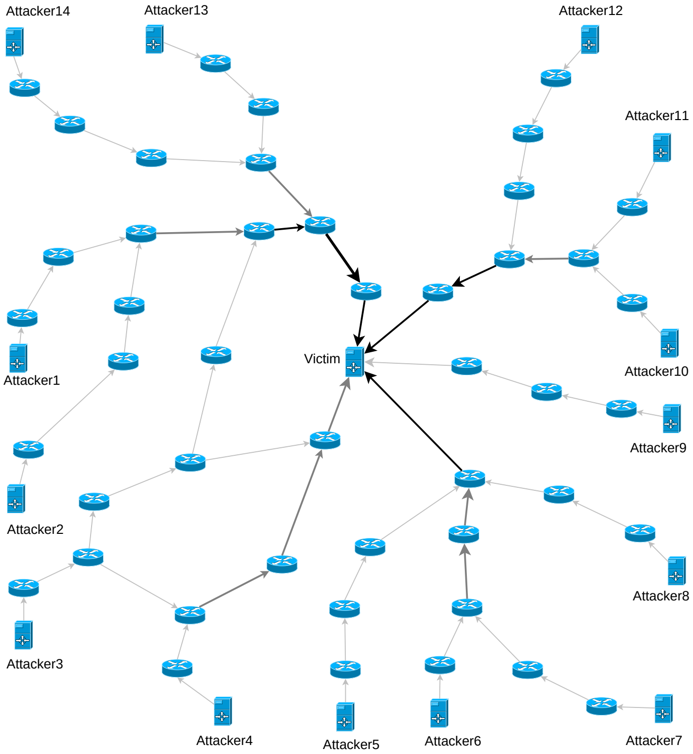

Figure 8: An example DDoS attack showing path merges

Table 2 presents the required average number of packets per attacker site to build a complete forward-paths graph for DDoS attacks of 100, 1000, 2000, 3000, 4000 and 5000 attacker sites. Remember that the victim site needs 20.23 packets on the average to construct the forward-paths graph in a DoS attack. In Table 2, the average number of packets decreases as the number of the attackers increases. The reason behind this is the fact that paths from multiple sources to a single destination merge toward the destination in the Internet. Hence, our technique requires less many packets to build a complete forward-paths graph for a DDoS attack.

Table 2: Average number of packets w.r.t. number of attacker sites in DDoS

Number of attacker sites	Average number of packets
100	20.12
1000	19.89
2000	18.64
3000	16.24
4000	13.62
5000	11.58

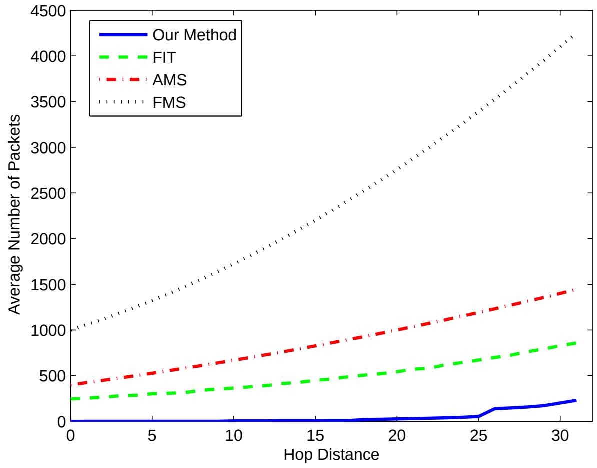

Figure 9: Our approach compared to FIT, AMS and FMS under DoS attacks

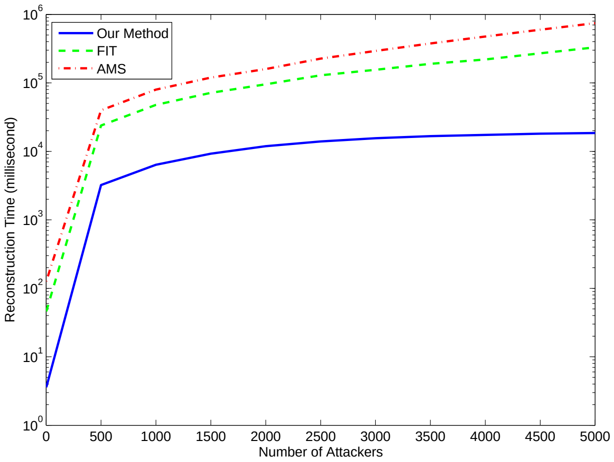

Figure 10: Our approach compared to FIT and AMS under DDoS attacks

In the following, we compare our results to other packet marking schemes: Fast Internet Traceback (FIT) [7], Authenticated Marking Scheme (AMS) [5] and Fragment Marking Scheme (FMS) [4]. We implemented FIT's 4/3 (n/n_{map}) scheme with 95% probability of reconstruction, AMS's $m > 5$ scheme with the marking probability 0.04 and FMS with the marking probability 0.04 as used in the corresponding papers. To achieve a fair comparison, we started the emulation at time zero for all methods where the victim site does not have any information about the topology. For each hop distance in iPlane dataset, we selected 5000 random path traces where the source of the trace is an attacker and the destination is the victim site. Then, we computed average number of required packets to build forward paths by repeating the experiment 1000 times. Figure 9 shows that FIT needs between 244 and 858 packets, AMS needs between 400 and 1400 packets and FMS needs between 1000 and 4100 packets on the average to construct forward paths for varying hop distances. On the other hand, our approach needs 1 to 229 packets on the average for the same path traces. The overall averages are 20.23, 457.18, 839.83 and 2253.38 for our approach, FIT, AMS and FMS respectively, regardless of the hop distance. On the other hand, these methods use two bytes (IP identification field) per packet and we use 40 bytes (record route field). As a result, our approach consumes more bandwidth compared to them (Section 6). In our implementation we represented path graphs by adjacency lists for all approaches because the IP-level Internet topology is sparse. Therefore, all approaches have complexity $O(|V| + |E|)$.

Next, we empirically compare the methods with respect to forward-paths graph construction time where the number of attackers vary from 1 to 5000 sampled at every 500 attackers. We assume that all routers, links and traffic loads in the network are identical and the attackers generate packets with the same rate, i.e., one packet per millisecond. Note that this experiment is to show the "relative" time differences among different approaches. In our experiment, we randomly chose the attackers regardless their hop distances to the victim. To improve our confidence, we repeated the experiment 100 times and computed the average time. Figure 10 shows the results of our experiment. In the figure, x-axis presents the number of attackers and y-axis presents the graph reconstruction time (ms) in log scale. Note that, we omitted FMS because the approach did not scale after 100 attackers. That is, FMS took several hours to construct the map even for 100 attackers. In the figure, the rate of the forward-path graph construction time decreases as the number of attackers increase due to the overlapping paths from the attackers toward the victim in our approach. The figure shows that our approach takes 21.42 milliseconds to 18.5 seconds to construct the graph for 1 to 5000 attackers. On the other hand, FIT and AMS require 46.53 milliseconds to 329.34 seconds and 156.18 millisecond to 745.56 seconds for the same 1 to 5000 attackers, respectively.

6. Discussions

Different from the previous probabilistic packet marking schemes, we exploited the IP record route field rather than the IP ID field of the IP protocol. The IP ID field is only two bytes whereas the record route field allows us to utilize 40 bytes. Therefore, our technique consumes more bandwidth compared to others.

Assuming that T is the IP traffic volume in bytes per second and l is the average IP packet length in bytes, $\frac{T}{l} = n$ denotes the number of packets per second. Holding n fixed, let T' be the traffic volume when we use an additional 40 bytes to construct forward paths. $T' = (l + 40) \times n$ will be the new traffic volume under our scheme. Then, $\frac{T' - T}{T} = \frac{40}{l}$ denotes our traffic overhead. It is reported that the IP packet length changes between 40 to 1500 bytes depending on the communication protocols and applications with a strong mode around 1300 bytes [30]. Using the mode as the reference packet length, our approach introduces $\frac{40}{1300} \approx 3.07\%$ additional traffic.

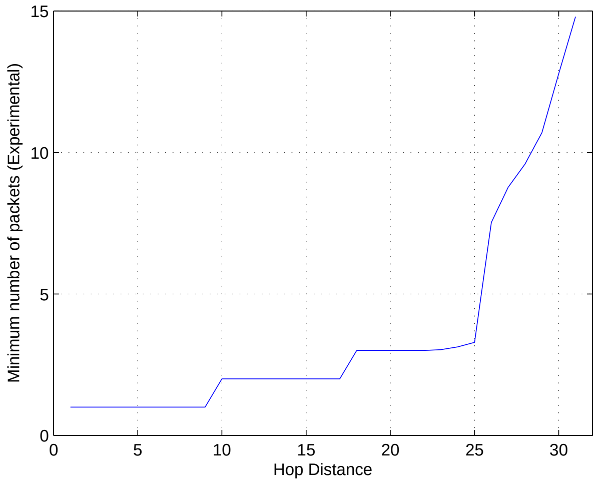

Figure 11: Minimum number of packets to construct a forward path (Experimental)

According to the Verizon's Data Breach Investigations Report in 2016 ”DoS attacks are either large in magnitude or they are long in duration, but typically not both” [31]. The report confirms that the majority of the DDoS attacks last from a few hours to 40 days or more. The report also states that a DDoS attack campaign generates 1.89 Mpps (million packets per second) on the average. Obviously, rogue packet rate per attack site depends on the total number of the attackers which can easily scale up to hundreds of thousands of compromised machines. As a result each attack site can generate a dozen packets per second on the average. Recently, hackers compromise IoT devices to conduct large scale DDoS attacks [32]. This new trend will increase the average size of botnets which in turn may reduce the number of packets per attack site.

In theory (Section 3.1), the minimum number of required *essentialRR* packets to construct a forward path is one for 1-9 hops, two for 10-17 hops, three for 18-25 hops and four for 26-31 hops. However, receiving only *essentialRRs* is very unlikely, e.g., for hop distance 20 the probability of immediately receiving all *essentialRRs* is 3.5×10^{-4} ($p^3 \times (1 - p)^{14}$ from Equation 1). To estimate the empirical minimum number of required packets, we selected 5000 random path traces where the source of the trace is an attacker and the destination is the victim site. Then, we computed minimum number of required packets to build forward paths by repeating the experiment 1000 times. Figure 11 shows the experimental minimum number of packets to construct forward paths. The figure follows the theoretical minimum number of packets up to hop length 25. Since the variance increases after 25 hops (Figure 5) the minimum number of packets changes between 7.53 to 14.8 for hops 26 to 31, on the average. Note that, Figure 11 captures the minimum even if it appears once out of 5000 path traces. Therefore, while Figure 11 gives us the minimum boundary, Figure 4 presents the realistic case in the long run. Obviously in case an attack site generates less many packets, constructing the forward paths becomes highly unlikely in our approach.

Finally, similar to other packet marking schemes, the proposed technique in this study requires support from router vendors and Internet Service Providers. In case the support is partial, the forward-paths graph that our algorithm generates would be partial. Nevertheless, a partial graph can still be beneficially to reduce the impact of DoS attacks and the variants in the Internet.

7. Conclusions

The Internet introduces a variety of vulnerabilities that puts the security and privacy of computer based systems under risk. One of the most perilous threats in the Internet is the Denial of Service (DoS) attack and the variants such as Distributed DoS (DDoS).

In this work, we proposed a novel probabilistic packet marking scheme to infer forward paths from attacker sites to a victim site to delegate the defense to the upstream Internet Service Providers (ISPs). We exploit the record route feature of the IP protocol to implement our probabilistic packet marking scheme. In our method, a router inserts its IP address in the record route options field of a packet with probability 1 as long as there is room. On the other hand, if there is no room in the options field, the router rewrites the options field with probability p or skips rewriting with probability $1-p$. We also presented an algorithm to construct a forward-paths graph from multiple attacker sites to a victim site. The algorithm starts from an empty forward-paths graph and gradually builds up the graph by incorporating the sub-paths reported in the record route options field of the received packets.

Our results show that a victim site can construct the forward path from an attacker site after receiving 20.23 packets on the average for DoS attacks. In DDoS attacks the average number of packets needed to construct a forward-paths graph decreases as the number of the attacker sites increases due to the overlapping paths toward the victim site. To illustrate, it is enough to receive 11.58 packets per attacker site on average to construct a forward-paths graph from 5000 attacker sites toward the victim site. Compared to the other techniques, our approach requires less many packets to construct the paths from attacker sites to a victim site. However, our technique consumes more bandwidth because it utilizes the 40 bytes record route options field of the IP header. Finally, source code of our technique is publicly available at our project website [2].

[1] A. Y. Nur, M. E. Tozal, ”Defending Cyber-Physical Systems against DoS Attacks,” IEEE International Conference on Smart Computing, SmartComp, May 2016

[2] Network Science Research Group, University of Louisiana at Lafayette - `http://nsrg.louisiana.edu/`

[3] H. Burch, B. Cheswick, ”Tracing anonymous packets to their approximate source,” in Proc. of 14th Systems Administration Conference, USENIX, December 2000

[4] S. Savage, D. Wetherall, A. Karlin, T. Anderson, ”Network support for IP traceback,” IEEE/ACM Transactions on Networking, vol. 9, no. 3, pp. 226-237, 2001

[5] D. Song, A. Perrig, ”Advanced and Authenticated Marking Schemes for IP traceback,” in Proc. IEEE INFOCOM, April 2001.

[6] D. Dean, M. Franklin, A. Stubblefield, ”An algebraic approach to IP traceback,” ACM Transactions on Information and System Security, May 2002

[7] A. Yaar, A. Perrig, D. Song, ”FIT: Fast Internet Traceback,” in Proc. IEEE INFOCOM, March 2005.

[8] A. Yaar, A. Perrig, D. Song, ”StackPi: New Packet Marking and Filtering Mechanisms for DDoS and IP Spoofing Defense,” IEEE Journal on Selected Areas in Communications, October 2006

[9] H. von Schelling, ”Coupon collecting for unequal probabilities,” American Mathematical Monthly, 61:306-311, 1954.

[10] iPlane Traceroute Dataset - 2015/06/01, http://iplane.cs.washington.edu/

[11] C. Gong, K. Sarac, ”IP traceback based on packet marking and logging,” in Proc. of IEEE International Conference on Communications, May 2005.

[12] A. Belenky, N. Ansari, ”IP traceback with deterministic packet marking,” IEEE Communications Letters, vol. 7, no. 4, pp. 162-164, April 2003.

[13] RFC 791 https://tools.ietf.org/html/rfc791

[14] Record Route for IPv6 (RR6) Hop-by-Hop Option Extension https://tools.ietf.org/html/draft-kitamura-ipv6-record-route-00

[15] B. Al-Duwairi, G. Manimaran, ”Novel Hybrid Schemes Employing Packet Marking and Logging for IP Traceback,” IEEE Trans. Parallel Distrib. Syst., vol. 17, no. 5, pp. 403- 418, May 2006

[16] J. Mirkovic, P. Reiher, ”A taxonomy of DDoS attack and DDoS defense mechanisms,” ACM SIGCOMM Computer Communication Review, vol. 34, no. 2, Apr. 2004

[17] V. A. Foroushani, A. N. Zincir-Heywood, ”TDFA: Traceback-based Defense against DDoS Flooding Attacks,” IEEE 28th International Conference on Advanced Information Networking and Applications, pp.597-604, May 2014.

[18] A. Belenky, N. Ansari, ”On Deterministic Packet Marking,” Computer Networks, vol. 51, no. 10, pp. 2677-2700, 2007

[19] M.T. Goodrich, ”Probabilistic Packet Marking for Large-Scale IP Traceback,” IEEE/ACM Trans. Networking, vol. 16, no. 1, pp. 15-24, Feb. 2008

[20] G. Carl, G. Kesidis, R. R. Brooks, and R. Suresh, ”Denial-of-service attack-detection techniques,” IEEE Internet Computing, vol. 10, pp. 82-89, 2006.

[21] Y. Xiang, K. Li, and W. Zhou, ”Low-Rate DDoS Attacks Detection and Traceback by Using New Information Metrics,” IEEE Trans. Information Forensics and Security, vol. 6, no. 2, pp.426-437, 2011

[22] S. T. Zargar, J. Joshi, and D. Tipper, ”A Survey of Defense Mechanisms Against Distributed Denial of Service(DDoS) Flooding Attacks,” IEEE Commun. Surveys and Tutorials, vol. 15, no. 4, pp.2046-2069, 2013

[23] S. Yu, W. Zhou, R. Doss, and W. Jia, ”Traceback of DDoS attacks using entropy variations,” IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 3, pp. 412-425, Mar. 2011

[24] E. Katz-Bassett, H. V. Madhyastha, V. K. Adhikari, C. Scott, J. Sherry, P. Van Wesep, T. Anderson, and A. Krishnamurthy, ”Reverse traceroute,” In Proc. of USENIX NSDI, 2010

[25] M. Sanchez, J. Otto, Z. Bischof, D. Choffnes, F. Bustamante, B. Krishnamurthy, and W. Willinger, ”Dasu: Pushing Experiments to the Internet's Edge,” In Proc. of USENIX NSDI, 2013

[26] R. Bush, O. Maennel, M. Roughan, and S. Uhlig, ”. Internet Optometry: Assessing the Broken Glasses in Internet Reachability,” In Proc. of ACM IMC, 2009

[27] W. W. T. Fok, X. Luo, R. K. P. Mok, W. Li, Y. Liu, E. W. W. Chan, and R. K. C. Chang, ”MonoScope: Automating network faults diagnosis based on active measurements,” in Proc. of IFIP IEEE, 2013

[28] D. Magoni, ”Tearing down the Internet,” IEEE Journal on Selected Areas in Communications, Aug. 2003

[29] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, A. Venkataramani, ”iPlane: An information plane for distributed services”, In Proc. of OSDI, Seattle, WA, 2006

[30] E. Garsva, N. Paulauskas, and G. Grazulevicius, ”Packet size distribution tendencies in computer network flows”, In Proceedings of 2015 Open Conference of Electrical, Electronic and Information Sciences (eStream 2015), IEEE 2015

[31] Verizon's Data Breach Investigations Report in 2016 http://www.verizonenterprise.com/verizon-insights-lab/dbir/2016/

[32] K. Angrishi, ”Turning Internet of Things(IoT) into Internet of Vulnerabilities (IoV) : IoT Botnets”, ArXiv e-prints, arXiv:1702.03681 https://arxiv.org/abs/1702.03681

[33] PlanetLab https://www.planet-lab.org/

[34] The Guardian, https://www.theguardian.com/technology/2016/oct/21/ddos-attack-dyn-internet-denial-service

Abdullah Yasin Nur is a Ph.D. candidate in the School of Computing and Informatics at the University of Louisiana at Lafayette. He received his M.Sc. degree in Computer Science from University of Louisiana at Lafayette in 2017. His research interests include network measurement, analysis and modeling, network security and graph theory.

Mehmet Engin Tozal is an Assistant Professor in the School of Computing and Informatics at the University of Louisiana at Lafayette and a member of the Informatics Program. He received his Ph.D. degree in Computer Science from the University of Texas at Dallas in 2012. His ongoing research involves analyzing, modeling and sampling real world complex systems including network topologies, social networks and information networks.