You Can Run, But You Can't Hide: An Effective Methodology to Traceback DDoS Attackers

K.T. Law
Department of Computer Science & Engineering
The Chinese University of Hong Kong
ktlaw@cse.cuhk.edu.hk

John C.S. Lui
Department of Computer Science & Engineering
The Chinese University of Hong Kong
cslui@cse.cuhk.edu.hk

David K.Y. Yau
Computer Science Department
Purdue University
West Lafayette, IN, 47907
yau@cs.purdue.edu

Abstract

With the increase of sophistication and severity of DDoS attack, it is important for a victim site to quickly identify the potential attackers and eliminate their traffic. Our work is based on the probabilistic marking algorithm by Savage[12] in which an attack graph can be constructed by a victim site. We extend the concept further such that we can deduce the local traffic rate of each router in the attack graph based on the received marked packets. Given the intensities of these local traffic rates, we can eliminate these attackers from sending high volume of traffic to a victim site. More importantly, we propose a theoretical method to determine the minimum stable time t_{min}, which is the minimum time it takes to accurately determine the local traffic rate of every participating router in the attack graph.

1. Introduction

Distributed denial-of service (DDoS) attack is a pressing problem on the Internet. Famous commercial sites such as Yahoo, Amazon and eBay were being attacked and were out of service for many hours due to the DDoS attack on February 2000[1]. In recent few years, DDoS attacks have increased in frequency, sophistication and severity. Recent study by the Computer Emergency Response Team (CERT) has indicated that the number of DDoS attacks have increased by 50% per year[7]. More alarming is the fact that many sophisticated DDoS attack tools, such as Tribal flood network (TBN), TFN2K and Trino can be easily downloaded from the Internet so users of these attack tools can simply launch a large scale attack with a few key strokes. One major difficulty to defend against DDoS attack is that attackers often use fake, or spoofed IP addresses as the IP source address. Therefore, attackers can easily disguise themselves as some other hosts on the Internet. Because of the stateless nature of the Internet, it is a difficult task to determine or trace the source of these attackers' packets and thereby locate the potential locations of these attackers. This is known as the DDoS attacker *traceback* problem.

An intrinsic characteristic of a denial-of-service attack is that source addresses of packets are usually spoofed. If all ISP's network administrators implement services such as ingress filtering[6], for example, refuse packet forwarding service whenever the packet's source IP address is not within the ISP's administrative domain, then these forged IP packets will not be able to enter the Internet core and thereby reducing the possibility of a DDoS attack. However, ingress filtering requires edge routers to have sufficient processing power, not only to inspect the packet's destination IP address for normal packet forwarding service, but also need to inspect the source address and determine whether it is a legitimate or illegitimate address. Another major problem about ingress filtering is that this technique is only effective if there is a widespread deployment in the networking community such that many ISPs are willing to deploy this service. Moreover, even with the enabling of ingress filtering service, attackers can still forge the source IP addresses as other hosts within their network domain. This makes eliminating attack traffic or even tracing back the origin of the attackers difficult. Alternative approach to DDoS traceback includes input debugging approach[16] which requires cooperation between system administrators of different ISPs. Therefore, it may not be able to trace the attackers in realtime or in the midst of a DDoS attack. Other approaches such as controlled flooding[3], which either generates many additional packets to the network (which can be viewed as another form of DDoS attack), or network logging[13], which requires additional storage and computational overhead of the participating routers. All in all, the above approaches have performance problems and signifi-

cant deployment difficulty.

Recently, Savage[12] proposed a probabilistic marking method so that participating routers can mark packets targeted to a victim site. Based on the received marked packets, the victim can construct an attack graph. However, one major shortcoming of this approach is that it does not identify and locate the potential attackers. In this work, we complement and enhance the probabilistic marking method by identifying the locations of these attackers. The contributions of our work are:

- We propose an effective method so that the victim site can estimate the local traffic rates of all participating routers in the attack graph.
- We provide a theoretical approach to determine the minimum stable time t_{min}, which is the *minimum* time it takes to accurately determine the local traffic rates. Note that the lower the value of t_{min} implies that we can determine the locations of attackers earlier.

The outline of the paper is as follows. In Section 2, we provide the necessary background of the probabilistic marking algorithm. In Section 3, we present the traceback method as well as how we can eliminate potential attackers for a simple but illustrative linear network. Experiments are presented in Section 4 to illustrate the effectiveness of our algorithms. Related work is presented in Section 5. Lastly, conclusion is given in Section 6.

2. Background

IP traceback is an approach to determine the source of an attack when a DDoS attack occurs. To accomplish this, whenever a packet traverses a router, the packet will be marked (either deterministically or probabilistically). The marked packets are filled with the partial or complete information of their respective traversed paths. Upon receiving these marked packets, a victim site can use the marking information to trace the attackers back toward the sources (e.g., the originated router of the attack traffic). In the following, we first present the necessary background of the probabilistic edge marking algorithm[12], which is used by a victim site to create an attack graph. Given the attack graph, we then present the methodology to estimate the local traffic rate of each router in the attack graph as well as the minimum time it takes for us to locate the potential attackers.

2.1. Probabilistic Edge Marking Algorithm

One way to implement an IP traceback service is to allocate enough space in an IP packet header so that one can use this space to record the traversed path of a packet. For example, each router, beside performing the normal packet forwarding and routing functions, also records or appends its own ID in the pre-allocated space at the packet's header so that when a victim receives a marked packet, victim can examine the packet's header and obtain the complete traverse path information of the marked packet. However, one major problem about this simple approach is that the length of a traversed path (e.g., number of hops) of a packet is not fixed. Therefore, it is impossible to pre-allocate sufficient amount of space in the packet's header in a prior fashion. Another technical difficulty of recording a complete path information of each packet to the victim is that an attacker can potentially manipulate this path information and fill in false router's identification in the packet's header so as to mislead the victim site.

Recently, probabilistic edge marking algorithm was proposed by Savage[12]. The idea is that instead of recording the complete path information of a packet, the goal is to record each traversed *edge* from the attacker to the victim site in a probabilistic fashion. Under the probabilistic edge marking algorithm, three finite fields are pre-allocated in the IP headers. These three fields are $\{$*start, end, distance*$\}$. The *start* and *end* fields store the IP addresses of the two routers at the end points of the marked edge while the *distance* field records the number of hops between the marked edge and the victim site. When a victim site is under a DDoS attack, the victim site will send a *"marking request signal"* to a set of routers[1] to participate in the probabilistic edge marking process and the participating routers will mark each packet targeted to the victim site with *probability* p. Whenever an packet which is targeted to the victim site passes through a router in the enabling set, the router, upon deciding the out-going edge of this packet through the standard routing table lookup, also marks the out-going edge to this packet's IP header with a probability p. In this case, the router records its IP address into the *start* field and sets the value of the *distance* field to zero. If the router decides not to mark the packet, the router checks whether the distance field is equal to zero. If it is equal to zero, the router records its IP address in the *end* field and then increments the *distance* field by one. If the distance field is not equal to zero, the router simply increments the *distance* field by one. Note that the mandatory increment of the distance field is crucial so as to minimize the probability of spoofing a marked edge. Any packet generated by an attacker will have distance greater than or equal to the hop count between the victim and the attacker. Therefore, a single attacker cannot forge any edge between himself and the victim. Figure 1 illustrates the probabilistic edge marking algorithm by each participating router.

Due to the property of the probabilistic marking algo-

[1] The set of routers can be all routers which are within $d \geq 1$ hops away from the victim site.

Marking procedure at router R:

for (each packet w targeted to the victim V) {
  generate a random number x between [0..1);
  if ($x < p$) /* *router R needs to mark the packet* */
    write R into w.start and 0 into w.distance;
  else /* *router R doesn't need to mark the packet* */
    if (w.distance == 0)
      write R into w.end;
    increment w.distance;

Figure 1. Probabilistic Edge Marking Algorithm for each participating router.

rithm, each traversed edge of an attacking packet will have a different probability of being marked or unmarked. Let $P_m(d)$ denote the probability that a victim site will find an edge which is d hops away as a marked edge. In general, we have

$$P_m(d) \quad = \quad p(1-p)^d \qquad\qquad d \geq 0. \tag{1}$$

In other words, an edge which is d hops away from the victim V will only be marked if a router which is connected to that edge decides to mark the packet and the remaining routers along the packet's traversed path decide not to mark (or reset the mark) this packet. Let $P_u(d)$ be the probability that a victim V will *not* find an edge which is d hops away as a marked edge. We have

$$P_u(d) \quad = \quad (1-p)^{d+1} \qquad\qquad d \geq 0. \tag{2}$$

In other words, all routers along the path to the victim decide not to mark the packet. Figure 2 illustrates the set of marked and unmarked edges collected by the victim V under a simple linear network topology. In this example, the victim V can collect 4 types of packets. Three of them are marked packets with marked edges (R_3, R_2), (R_2, R_1) and $(R_1, -)$. Also, the victim V can also receive unmarked packets.

A victim V, upon receiving packets, needs to first filter out those unmarked packets (since they don't carry any information in the attack graph construction). For all the collected marked packets, the victim needs to execute the *graph construction* algorithm so as to re-construct the attack graph. Figure 3 illustrates the attack graph construction algorithm.

One major shortcoming of the probabilistic edge marking algorithm [12, 14] is that it does not provide an effective mean to *locate* the positions of potential attackers. In general, the attack graph only provides the topology of traffic which was targeted to a victim site. In the following, we present an effective methodology to locate the potential attackers.

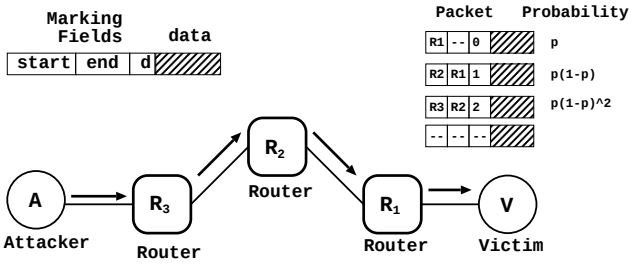

Figure 2. Example of a Probabilistic Edge Marking.

Attack Graph Construction Procedure at victim V

let $\mathcal{G}$ be a tree with root being victim V;
let edges in $\mathcal{G}$ be tuples(start,end,distance);
for (each received marked packet w) {
  if (w.distance==0) then
    insert edge (w.start,V,0) into $\mathcal{G}$;
  else
    insert edge (w.start,w.end,w.distance) into $\mathcal{G}$;
}
remove any edge (x, y, d) with $d \neq$ distance
        from x to V in $\mathcal{G}$;
extract path $(R_i..R_j)$ by enumerating acyclic paths
        in $\mathcal{G}$;

Figure 3. Attack Graph Construction Algorithm.

3. Attacker Traceback

The marked packets collected by a victim can serve two purposes. First, they are used to create an attack graph $\mathcal{G}$. Secondly, one can also use these marked packets to generate *statistical information* so as to traceback the potential attackers. We will show that based on the collected marked packets, we can deduce the *local traffic rate* for every router in the attack graph $\mathcal{G}$ wherein local traffic of router R_i is defined as the traffic generated from the local network domain of R_i to the victim site V. Based on the traffic intensity of each of the local traffic rate at each router, we can determine the possible location of the attacker. It is important to point out that one major technical difficulty in estimating the local traffic rates is to determine the *minimum stable time* t_{min}, which is the minimum time we need to collect the marked packets so that the calculated local traffic rates are *correct* and *stable*. In the following two subsections, we illustrate how we can determine the local traffic rates and the minimum stable time.

3.1. Determination of Local Traffic Rates and Minimum Stable Time t_{min}

To illustrate the methodology, let us consider the following illustrative example. Figure 4 illustrates a linear topology with d routers R_i, where $1 \leq i \leq d$. Router R_i receives its local traffic (e.g, traffic which is generated from its local network domain) to the victim V. Let λ_i denotes the average local traffic rate, in unit of packets per second, from the router R_i to the victim V. Each router also performs packet routing/forwarding for all upstream traffics which are targeted to the victim V. For example, router R_i in Figure 4 performs packet routing/forward for traffic λ_j for $j = i, i+1, \ldots, d$. The main idea is that if one can *estimate* the local traffic rates λ_i for all these routers from the received marked packets, then based on the intensity of these traffic rates, one can identify the location of the attacker.

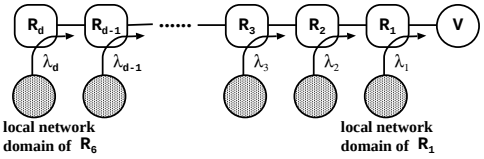

Figure 4. Linear Topology with local traffic rates λ_i to the victim V.

Due to the nature of the probabilistic marking, each router may mark any local or upstream transit packet to the victim V. If the router R_j decides to mark a transit packet, then it resets the packet's marking, if there is any, by any upstream routers of R_j. Let $\tilde{N}_j(t)$ be the random variable denoting the number of marked packets received by the victim V at time t such that the *start* field is equal to router R_j, for $1 \leq j \leq d$.To compare these random variables $\tilde{N}_j(t)$ for $1 \leq j \leq d$, we need the following definition from[11].

Definition 1 *Let X, Y be two random variables. We say X is stochastically larger than Y, denote as $X \geq_{st} Y$, if*

$$Prob[X > a] \geq Prob[Y > a] \quad \forall a.$$

It is clear that if we collect marked packets for a sufficiently long period, then we have the following relationship:

$$\tilde{N}_j(t) \quad \geq_{st} \quad \tilde{N}_{j+1}(t) \qquad \text{for } j = 1, \ldots, d-1. \quad (3)$$

The above relationship holds because

1. Each router carries its local traffic and upstream transit traffic to the victim V. In a long run, the number of *marked* packets by R_j is greater than the number of *marked* packets by R_{j+1}.
2. Each router R_j will probabilistically mark a transit packet to the victim V. Therefore, the router R_j erases any edge marking by a upstream router.

Let $N_j(t)$ be the number of marked packets received by the victim V at time t such that the *start* field is equal to router R_j. We have the following relationship:

$$N_j(t) \quad = \quad \left(\sum_{i=j}^{d} \lambda_i t \right) p(1-p)^{j-1} \quad \text{for } j = 1, \ldots, d. \quad (4)$$

The first term corresponds to the aggregated number of packets targeted to victim V at time t, the remaining term is the probability that these packets are marked by R_j and that these packets are not marked by any downstream routers of R_j. Equation (4) is a system of triangular equations, we can arrange the above equations and obtain the local traffic rates λ_i at each router R_i as:

$$\lambda_i \quad = \quad \begin{cases} \frac{N_d(t)}{tp(1-p)^{d-1}} & i = d \\ \frac{N_i(t)}{tp(1-p)^{i-1}} - \frac{N_{i+1}(t)}{tp(1-p)^{i}} & 1 \leq i \leq d-1. \end{cases} \quad (5)$$

The remaining technical issue is, to have an accurate estimation of these local traffic rates, we have to make sure that the conditions in Equation (3) are satisfied. It is not difficult to observe that these conditions are not satisfied, for example, when the duration of collecting these marked packet is very small (e.g $t \approx 0$). In the following, we provide an analytical method to derive the minimum stable time t_{min} such that the conditions of Equation (3) are satisfied. Once these conditions are satisfied, we can then use the estimate of the local traffic rates based on Equation (5) to traceback the potential attackers.

To simplify the notation, let us define:

$$\tilde{\lambda}_j \quad = \quad \left(\sum_{i=j}^{d} \lambda_i \right) p(1-p)^{j-1} \quad \text{for } j = 1, 2, \ldots, d \quad (6)$$

where d is the number of hops from the victim V to the furthest routers in the attack graph. Assuming that the random variable $\tilde{N}_j$ and $\tilde{N}_{j+1}$ are Poisson random variables, we can re-formulate the problem of finding the minimum stable time t_{min} such that:

$$\text{Prob}[\tilde{N}_j(t_{min}) \geq \tilde{N}_{j+1}(t_{min})] \;\geq\; p_{threshold} \quad \forall j \in \{1, ..d\} (7)$$

where $p_{threshold}$ is a large probability (e.g., $p_{threshold} = 0.95$). In other words, we want to find the minimum time t_{min} such that with a very high probability, the number of collected marked packets for router R_{j-1} is higher than the number of collected marked packets for router R_j for $1 \leq j \leq d$. Since the random variables $\tilde{N}_j(t)$ and $\tilde{N}_{j+1}(t)$ are Poisson random variables, we have

$$\text{Prob}[\tilde{N}_j(t_{min}) \geq \tilde{N}_{j+1}(t_{min})] \quad =$$

$$\sum_{k=0}^{\infty} \text{Prob}[\tilde{N}_j(t_{min}) \geq k] \text{Prob}[\tilde{N}_{j+1}(t_{min}) = k] =$$

$$\sum_{k=0}^{\infty} \left[\sum_{n=k}^{\infty} \frac{(\tilde{\lambda}_j t_{min})^n}{n!} e^{(-\tilde{\lambda}_j t_{min})} \right] \frac{(\tilde{\lambda}_{j+1} t_{min})^k}{k!} e^{(-\tilde{\lambda}_{j+1} t_{min})}$$

$$\forall j \in \{1, \ldots, d\}. \quad (8)$$

Based on the above expression, we can easily determine the minimum stable time t_{min} using some standard numerical methods[4]. Figure 5 illustrates the *"local traffic estimation procedure"* to estimate the local traffic rate of each router in an attack graph $\mathcal{G}$.

Local traffic rates estimation procedure at victim V:

let $\mathcal{P}$ be a linear path to the victim V;
let $\mathcal{S}_{\mathcal{P}}$ be the set of routers along the path $\mathcal{P}$;
/ Note that $\mathcal{P}$ and $\mathcal{S}_{\mathcal{P}}$ are derived from */*
/ the probabilistic edge marking algorithm. */*
set stable = false;
while (stable == false){ */* has not reached t_{min} yet */*
  collect marked packets for some time;
  for (each router R_i in $\mathcal{S}_{\mathcal{P}}$) {
    calculate the local traffic λ_i of R_i based on
      Equation (5);
  }
  determine whether we have reached the minimum
    stable time t_{min} or not for each router in $\mathcal{S}_{\mathcal{P}}$
      based on Equation (8);
  if (t_{min} is reached)
    stable = true;
}
output: local traffic rates λ_i for router R_i in $\mathcal{S}_{\mathcal{P}}$.

Figure 5. Algorithm to determine local traffic rates for every router in the attack graph.

3.2. Elimination of Attackers

In the previous subsection, we have presented the methodology on how one can estimate the local traffic rate for each router in the attack graph. In this subsection, we present the algorithm to find the potential attackers and eliminate their attack traffics.

Given a linear path $\mathcal{P}$ in the attack graph, the victim V can choose to reduce its traffic by a pre-determined fraction $\mathcal{T}_{cutoff}(\mathcal{P})$. To reduce the traffic, the victim V determines the local traffic rates for all routers in the path $\mathcal{P}$, then it sorts these traffic rates in an non-increasing order. The victim V then sends signal to a subset of routers along the path $\mathcal{P}$ and instructs these routers to stop forward their respective local traffics to the victim V. Figure 6 illustrates the procedure to eliminate the attacker's traffic to victim V. We refer readers to [8] for the illustration of algorithm for determining local traffic intensities so as to locate potential attackers.

Eliminate Potential Attackers along path $\mathcal{P}$

Let $\mathcal{P}$ be a linear path to the victim V;
Let $\mathcal{S}_{\mathcal{P}}$ be the set of routers along the path $\mathcal{P}$;
Derive the local traffic rates λ_i for router $R_i \in \mathcal{S}_{\mathcal{P}}$ based
    on the "Local Traffic Estimation procedure";
Sort $\{\lambda_i\}$ in non-increasing order and let the
    sorted sequence be $\{\acute{\lambda}_i\}$;
Find the *"minimum"* i^* such that $\frac{\sum_{j=1}^{i^*} \acute{\lambda}_j}{\sum_{k=1}^{d} \lambda_k} \geq \mathcal{T}_{cutoff}(\mathcal{P})$;
Send control signals to routers $\{R_j\}_{1 \leq j \leq i^*}$ so that routers
    can stop their local traffic to victim V;

Figure 6. Algorithm to find the potential attackers and eliminate their attack traffics.

4. Experiments

In this section, we perform experiments to illustrate the effectiveness in locating the attackers. In particular, we show the correctness of using Equation (8) to find the minimum stable time t_{min} such that we can compute the intensities of the local traffic rates for all routers in $\mathcal{G}$. Note that a smaller value of t_{min} implies that we can find the location of the attackers earlier. We also illustrate various factors which can influence the values of the minimum stable time t_{min}. Unless we state otherwise, the experiments we carry out are based on the linear topology depicted in Figure 7. To derive the attach graph, the victim V informs the participating routers to mark packets with probability $p = 1/25$. To estimate whether we have reached the stability conditions specified by Equation (8), we set $p_{threshold} = 0.95$. Normal local traffic from each router is generated based on a Poisson process with an average rate of 100 pkt/sec. We define a variable, **Attack Traffic Ratio** (ATR), which is the ratio of the attack average traffic rate to the normal average local traffic. For example, if the normal average local traffic is 100 pkts/sec and ATR is set to 20, then the attack average traffic rate is equal to 2000 pkts/sec.

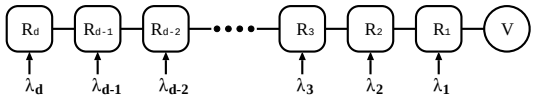

Figure 7. Network Topology for Exp. 1 and 2.

Experiment 1: (Correctness of Using Equation (8) to estimate the minimum stable time t_{min} and the performance tradeoff of $p_{threshold}$). In Experiment 1.A, we use

a linear topology in Figure 7 with 25 routers (e.g., $d = 25$). There is one attacker and he is located at the furthest router R_{25} and the ATR is 20. The normal traffic and the attacker traffic is generated using three methods: (1) Poisson process, (2) constant packet rate (e.g., a rate of 100 pkt/sec implies that every 0.01 second, there is a new packet generated by a router), (3) burst rate (e.g., a rate of 100 pkt/sec implies that we generate 100 pkts in one burst for every second). To compute the minimum stable time t_{min}, we use the mathematical derivation in Equation (8). Table 4 illustrates that different values of t_{min}, our theoretical computed $p_{threshold}$ value is very close to the simulated $p_{threhold}$ values for different traffic generation processes. This indicates that we can accurately estimate the minimum stable time t_{min} for different traffic arrival processes. This indicates that our method is very robust and accurate in estimating the local traffic rates.

Table 1. Minimum Stable Time: Theoretical vs. Simulation Results for different packet generation processes.

t_{min} (sec)	$p_{threshold}$ (%)			
	Theoretical Computed Values	Simulation		
		Poisson	Constant	Burst
5.0	79.46	79.37	79.69	79.88
10.0	87.35	87.36	87.36	87.32
15.0	91.80	91.74	91.78	91.84
20.0	94.55	93.93	94.48	94.59
25.0	96.31	96.19	96.21	96.54

Equation (8) indicates that if the stochastic relationship in Equation (3) are satisfied with a high probability $p_{threshold}$, the estimation of local traffic will become accurate. The accuracy can be expressed by the variance of the real traffic rates.

In Experiment 1.B, we illustrate how the parameter $p_{threshold}$ affects the quality of the estimated minimum stable time t_{min}. Figure 8 illustrates that for different values of $p_{threshold}$, we have different values of t_{min} and its associated variance. Figure 8 shows that the variance of the minimum time estimation approaches to zero (e.g, which represents a highly accurate estimation of t_{min}) when we set $p_{threshold} \geq 0.8$. Of course, a higher value of $p_{threshold}$ also implies a larger value of t_{min}, which can affect how quickly we can determine the location of the attackers. Nevertheless, because t_{min} is less than 60 seconds, this implies that we can quickly response to the DDoS attack and locate the attackers.

Experiment 2: (Effect of the network's diameter, number of attackers and attack traffic intensity to the minimum stable time t_{min}). In this experiment, we illustrate how the length of an attack path, the number of attackers, and the attack traffic ratio ATR can affect the values of the minimum stable time t_{min}.

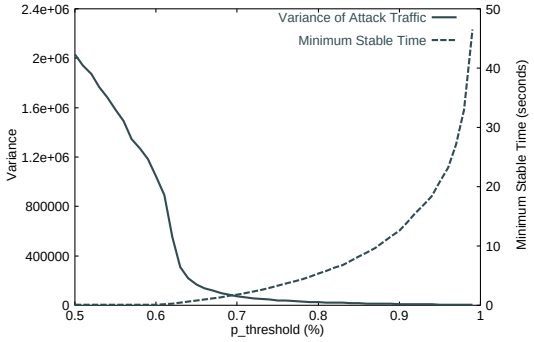

Figure 8. Relation between Variance of Attack Traffic and Minimum Stable Time.

In Experiment 2.A, we study the relationship between the length of an attack path and the minimum stable time. We use a linear network topology with varying length of attack path and compute the minimum stable time t_{min}. Assume an attacker locates at the *furthest* router (e.g., R_d) and ATR is equal to 20. Figure 9 shows that as the length of an attack path increases, the minimum stable time t_{min} also increases. The reason for this linear relationship is that the victim V needs to take a longer time to collect sufficient number of *marked* packets from the further router.

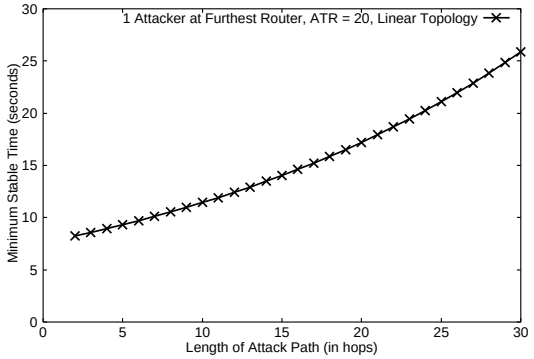

Figure 9. Relationship between Length of Attack Path and Minimum Stable Time t_{min}.

In Experiment 2.B, we study how the number of attackers and their relative positions can influence the value of the minimum stable time t_{min}. We consider three attackers in a linear network topology and these three attackers are distributed by three different configurations. For configuration I, these three attackers are located in routers R_1, R_2 and

R_3 (e.g., the 3 closest routers to the victim V). For configuration II, these three attackers are located in the three furthest routers from the victim V. In configuration III, these three attackers are *evenly distributed* around the linear network, for example, they are located in routers $R_{\lfloor \frac{d*i}{3} \rfloor}$ where d is the length of attack path and i=1,2,3. Figure 10 illustrates the minimum stable time when ATR$= 20$ and $p_{threshold} = 0.95$. We observe that when the attackers are closer to the victim (e.g, configuration I), the achieved minimum stable time is lower than other configurations. The minimum stable time t_{min} achieves highest value when attackers are evenly distributed in the network (e.g, configuration III). The reason for this ordering is because the estimation of local traffic rates at the furthest router takes longer time to become stable while the estimation of local traffic rates at the nearest router takes less time. Regardless of the distribution of the attackers' locations, we can determine their locations within 30 seconds, which shows the effectiveness of the proposed methodology.

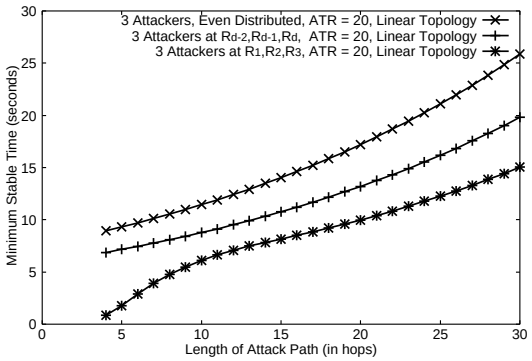

Figure 10. Relationship between Length of Attack Path and Minimum Stable Time t_{min} for different relative positions.

5. Related Work

One weakness of the IP protocol is that the source address can be filled by the hosts [2]. There is no provision to discover the true origin of the packet. Various researchers have proposed different approaches to solve the attack problem, including throttle technique[18].

While ingress filtering can minimize the number of attack packets, the best way to address the denial-of-service attack is to traceback the attackers and stop their attacks. One of the traceback approaches is called the link testing. Link testing is to interactively test its upstream link starting from the closet routers. This process is repeated recursively until the source of attack is found. There are many ways to implement the link testing and we explain two of them: input debugging and controlled flooding. For input debugging, the victim develops attack signature based on the characteristics of the attack packets. Then victim requests network administrators to filter attack packets on some egress port and identifies which ingress port they arrived from. The procedure is repeated recursively and the origin of the attack can be found. This approach is usually implemented manually and Stone[16] proposed automatic way to trace the attacker within their own networks. But the communication overhead for exchanging message is very high and operations are constrained by different network administrations. Burch and Cheswick[3] proposed another approach which is called controlled flooding. Victim interactively floods its upstream links starting from the closet routers. Then the victim observes the changes in the rate of attack packets and identifies which link the attack packets came from. Selective flooding is repeated to further routers until the full attack path is found. Controlled flooding consumes many network resource such as network bandwidth and can be considered as another form of attack.

Both link testing approaches require the attack remains active during traceback process. Logging is another approach to traceback even after the attack has completed. Partial packet information will be stored in routers for the traversed packets and this information provides a trail of the attack path. Full attack path can be reconstructed by applying some data extraction methods. The advantage of this approach is that it does not increase traffic flow, but it increases the storage requirement of the participating routers. Snoeren *et al.*[13] proposed efficient hash-based technique to store the packet digestion, and stated the storage requirement is approximately below one percentage of the link capacity per unit time. However, because tens of thousands of packets can traverse a router each second, the logging data can still grow quickly to an enormous size and this is especially true for high speed link. Therefore, database integration management, processing and storage overhead of routers would be a problem.

While logging requires huge amount of space in each router, Bellovin[15] proposed ICMP traceback (and later extensions by Wu *et al.*[17, 9]) which can traceback the attackers without incurring much overhead on the routers. The routers probabilistically generates an authenticated copy of a packet, including information about the adjacent routers along the path to the destination. These information can be used to reconstruct the path to the attackers. However, ICMP traffic is different from the normal traffic and may be blocked or rate limited. It also needs key distribution management to deal with the faking ICMP packet problem.

Savage *et al.*[12] proposed probabilistic marking for traceback without generating separate ICMP packets to the

victim. Routers mark packets probabilistically and store the partial path information in the IP header. Each piece of information represents a sample edge of the attack path. Victim collects the attack packets and can reconstruct the attack path based on the partial path information. This approach does not need the coordination among the network administrators and it does not increase the traffic flow or the storage requirement of a router. Lee and Park[10] analyzed this marking approach and pointed out that spoofing of the marking field may impede traceback by the victim. Attackers may choose the spoofed marking value, source address to hide themselves. Dean *et al.*[5] formulated the traceback problem as a polynomial reconstruction problem, They used algebraic coding theory to encode traceback information in the packet, similar to Savage approach. It also suffers the same spoofing problem and may be more vulnerable without the distance field in the marking. Song and Perrig[14] reported that if the victim knows the map of its upstream routers, it does not need the full IP address in the packet marking. They improved Savage's marking approach by hashing so as to achieve a lower false positive rate and a lower computation overhead. They proposed efficient authentication of packet markings to filter packets with spoofed markings from the attackers. Note that approaches by Savage and Song[12, 14] provide the topology of the attack graph. Our approach can be view as a complementary approach to their so as to locate potential attackers in the attack graph.

6. Conclusion

In this paper, we consider the traceback problem during a DDoS attack. Instead of dealing with the issue of *detecting* a DDoS attack, we address how we can locate and eliminate potential attackers. Our approach uses the probabilistic marking algorithm: a victim site V, upon discovering that it is being attacked, will request a set of routers to mark all packets target to V. Based on the collected marked packets, V can then construct an attack graph. We enhance the probabilistic marking algorithm by determining the local traffic rate of each router in the attack graph. Based on the traffic intensities, we can send signal to the corresponding routers to eliminate the attack traffic. One important technical contribution we made is that we provide a theoretical approach to determine the minimum stable time t_{min}, which is the minimum time it takes to accurately determine the local traffic rate of every participating router in the attack graph so that we can locate the potential attackers. We carried out experiments to illustrate the effectiveness and robustness of our algorithms. We showed that we can locate attackers within a short duration. We believe this is a valuable first step towards an automated network-wide traceback facility.

References

[1] "computer emergency response team, cert advisory ca-2000-01: Denial-of-service developments". http://www.cert.org/advisories/ca-2000-01.html. 2000.

[2] S. Bellowin. Security problems in the tcp/ip protocol suite. pages 32 – 48.

[3] H. Burch and B. Cheswick. Tracing anonymous packets to their approximate source. In *Usenix LISA*, December 2000.

[4] R. L. Burden and J. D. Faires. *Numerical Analysis*. PWS-Kent Publishing Company, Boston, 1988.

[5] D. Dean, M. Franklin, and A. Stubblefield. An algebraic approach to ip traceback. In *Proceedings of Network and Distributed System Security Symposium, NDSS '01*, February 2001.

[6] P. Ferguson and D. Senie. Rfc 2267: Network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing. *The Internet Society*, January 1998.

[7] J. Howard. An analysis of security incidents on the internet. *PhD Thesis, Carnegie Mellon University*, August 1998.

[8] K. T. Law, J. C. S. Lui, and D. K. Y. Yau. You can run, but you can't hide: An effective methodology to traceback ddos attackers. In *Technical Report of CSE Department at the CUHK, CS-TR-2002-06*.

[9] A. Mankin, D. Massey, C.-L. Wu, S. F. Wu, and L. Zhang. On design and evaluation of intention-driven icmp traceback. In *Proceedings of IEEE International Conference on Computer Communications and Networks*, 2001.

[10] K. Park and H. Lee. On the effectiveness of probabilistic packet marking for ip traceback under denial of service attack. In *Proceedings of IEEE INFOCOM '01*, pages 338 – 347, 2001.

[11] S. M. Ross. *Stochastic Processes*. John Wiley Series, New York, 1996.

[12] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for ip traceback. In *Proceedings of the 2000 ACM SIGCOMM Conference*, pages 295 – 306, Stockholm, Sweden, August 2000.

[13] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer. Hash-based ip traceback. In *Proceedings of the ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, August 2001.

[14] D. X. Song and A. Perrig. Advanced and authenticated marking schemes for ip traceback. In *Proceedings of IEEE INFOCOM '01*, April 2001.

[15] E. Steven M. Bellovin. Icmp traceback messages, internet draft: draft-bellovin-itrace-00.txt. submitted Mar. 2000, expiration date Sep. 2000.

[16] R. Stone. Centertrack: An ip overlay network for tracking dos floods. In *Proceedings of 9th USENIX Security Symposium*, August 2000.

[17] S. F. Wu, L. Zhang, D. Massey, and A. Mankin. Intention-driven icmp trace-back, internet draft: draft-wu-itrace-intention-00.txt. submission date Feb. 2001, expiration date Aug. 2001.

[18] D. K. Y. Yau, J. C. S. Lui, and F. Liang. Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles. In *IEEE International Workshop on Quality of Service (IWQoS)*, May 2002.