You Can Run, But You Can't Hide: An Effective Statistical Methodology to Trace Back DDoS Attackers

Terence K.T. Law, John C.S. Lui, *Senior Member*, *IEEE*, and David K.Y. Yau, *Member*, *IEEE Computer Society*

Abstract—There is currently an urgent need for effective solutions against distributed denial-of-service (DDoS) attacks directed at many well-known Web sites. Because of increased sophistication and severity of these attacks, the system administrator of a victim site needs to quickly and accurately identify the probable attackers and eliminate the attack traffic. Our work is based on a probabilistic marking algorithm in which an attack graph can be constructed by a victim site. We extend the basic concept such that one can quickly and efficiently deduce the intensity of the "local traffic" generated at each router in the attack graph based on the volume of received marked packets at the victim site. Given the intensities of these local traffic rates, we can rank the local traffic and identify the network domains generating most of the attack traffic. We present our traceback and attacker identification algorithms. We also provide a theoretical framework to determine the minimum stable time t_{min}, which is the minimum time needed to accurately determine the locations of attackers and local traffic rates of participating routers in the attack graph. Entensive experiments are carried out to illustrate that one can accurately determine the minimum stable time t_{min} and, at the same time, determine the location of attackers under various threshold parameters, network diameters, attack traffic distributions, on/off patterns, and network traffic conditions.

Index Terms—DDoS attack, traceback, attack traffic filtering, minimum stable time.

1 Introduction

The Internet is an open architecture susceptible to various forms of network attack, a prime example of which is the distributed denial-of-service (DDoS) attack. Well established commercial sites such as Yahoo, Amazon, and eBay were being attacked and taken out of service for many hours in February 2000 [2]. Recently, DDoS attacks have increased in frequency, sophistication, and severity. A recent study by the Computer Emergency Response Team (CERT) indicates that the number of DDoS attacks has increased by 50 percent per year [8]. 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 such that users of these tools can launch a large-scale site attack with a few simple key strokes.

A major difficulty in defending against DDoS attacks is that attackers can easily disguise themselves by *spoofing* the source IP addresses of their packets. Together with the stateless nature of the Internet, it is then very difficult to deduce and determine the originating sources of the attack packets. This is the DDoS attacker *traceback* problem.

To prevent spoofed packets from entering the Internet, an ISP network administrator can implement services such as ingress filtering [7]. If packets enter an edge router different from the known attached network domain, they can be dropped. However, ingress filtering requires edge routers to have sufficient processing power to inspect the source IP address of every packet and determine whether the address is legitimate or not. More importantly, ingress filtering requires *universal deployment* at the network edge to eliminate all spoofed packets.

Other methods to counter DDoS attacks have been proposed but all have significant performance and deployment problems. For example, the *input debugging* technique [19] requires cooperation between system administrators of different ISPs. The major limitation of this approach is that it may not be able to trace back the DDoS attackers in real time or while a DDoS attack is ongoing. *Controlled flooding* [4] generates many additional packets in the network, which can itself lead to a form of DDoS attack. *Network logging* [16] requires significant additional storage and computational overhead at the participating routers.

Various researchers [12], [14], [15], [17] have proposed to trace back DDoS attackers using a probabilistic packet marking algorithm. In these solutions, each participating router will probabilistically mark packets destined for a victim site. Based on the received marked packets, the victim can construct an attack graph of paths traversed by these packets until they reach the victim. The constructed attack graph consists of the paths traversed by the attack packets as well as the paths traversed by the regular packets. Therefore, it is not necessarily effective in isolating all the responsible attackers.

- *T.K.T. Law and J.C.S. Lui are with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Shatin, N.T., Hong Kong.*
  E-mail: ktlaw@alumni.cuhk.net, cslui@cse.cuhk.edu.hk.
- *D.K.Y. Yau is with the Department of Computer Sciences, Purdue University, 250 N. University St., West Lafayette, IN 47907-2066.*
  E-mail: yau@cs.purdue.edu.

Manuscript received 2 Jan. 2002; revised 26 Jan. 2004; accepted 18 Nov. 2004; published online 22 Aug. 2005.
For information on obtaining reprints of this article, please send e-mail to:

In this paper, we complement and enhance the probabilistic marking method by more accurately identifying the locations of the DDoS attackers. Our work is as follows:

1. We propose an effective method to allow a victim site to deduce the *local traffic rates* of all participating routers in an attack graph.
2. We provide a theoretical framework to determine the minimum stable time t_{min}, which is the *minimum* time required to accurately determine the local traffic rates of all the participating routers. Note that the lower the value of t_{min}, the earlier a system administrator can determine the locations of the attackers.
3. The traceback methodology is effective for a general network topology and different attack patterns.

The balance of the paper is organized as follows: In Section 2, we provide the necessary background of the probabilistic marking algorithm for our work. In Section 3, we introduce our traceback method and discuss how we can isolate potential attackers for a simple but illustrative linear network topology. In Section 4, we extend our traceback method to a general network topology. Experiments are presented in Section 5 to illustrate the effectiveness of our traceback method. Related work and implementation issues are discussed in Section 6. Section 7 concludes.

2 Background

IP traceback is an approach to determine the sources of a DDoS attack. To accomplish the goal, routers are assumed to provide additional *packet marking* functions besides basic packet routing and forwarding. In particular, whenever a packet passes through a router, the packet will be marked, either deterministically or probabilistically. Marked packets have information about their respective traversed paths. Upon receiving the marked packets, a victim site, denoted by $\mathcal{V}$, can use the marking information to create an attack graph. Based on the attack graph and received packets, $\mathcal{V}$ can trace the attackers back towards the sources (i.e.., originators of the attack traffic). In the following, we first present the necessary background of the probabilistic edge marking algorithm in [14], which is used by the victim site $\mathcal{V}$ to create an attack graph. Given the attack graph, we then present a methodology to estimate the local traffic of all routers in the graph and the minimum time it takes to locate the potential attackers.

2.1 Probabilistic Edge Marking Algorithm

To support IP traceback, one needs to allocate sufficient space in the IP header to record the traversed path of a packet. For example, every router in the path can append its own IP address in the header. A major problem with this simple approach is that the length of the path (i.e., the number of hops) traversed by packet is not fixed. For example, one packet may take two hops to the victim site $\mathcal{V}$ whereas another packet may take 15. Therefore, it is impossible to preallocate a sufficient amount of space in the packet header. Another technical difficulty in recording the complete traversed path is that an attacker can potentially manipulate this path information to fill in false router identifications, in order to mislead the path analysis.

Algorithm: Edge Marking Procedure at router R

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

Fig. 1. Probabilistic edge marking algorithm for each participating router.

Rather than record the complete path, probabilistic edge marking [14], [17] proposes to record only a traversed *edge* from the attacker to the victim site $\mathcal{V}$ in a probabilistic fashion. The marking algorithm uses some unused fields in the existing IP header to store three fields of information. The 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 $\mathcal{V}$. When a victim site $\mathcal{V}$ is under a DDoS attack, it will send a `marking-request-signal` to a set of routers (i.e., all the routers which are within $d \geq 1$ hops from $\mathcal{V}$) requesting their participation in the probabilistic edge marking process. Each participating router will then mark each packet destined for $\mathcal{V}$ with *probability* p. In other words, whenever an IP packet addressed to $\mathcal{V}$ passes through a router in the enabled router set, the router, upon deciding the out-going edge of the packet through standard routing lookup, will mark the out-going edge in the packet's IP header with probability p. If marking is performed, the router records its IP address in the `start` field and sets the value of the `distance` field to zero. If the router decides not to mark the packet, the router needs to check whether the `distance` field of the packet is equal to zero or not. 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 because it is used to minimize the probability of spoofing a marked edge. Under the marking method, any packet generated by an attacker will have a distance greater than or equal to the hop count between the victim site $\mathcal{V}$ and the attacker. Therefore, an attacker cannot forge any edge between itself and $\mathcal{V}$. The probabilistic edge marking algorithm used by each participating router in the enabled set is illustrated in Fig. 1.

By the property of the probabilistic marking algorithm, each traversed edge of an attack packet will have a different probability of being marked or unmarked. Let $P_m(d)$ denote the probability that a victim site $\mathcal{V}$ will find an edge which is d hops away as a marked edge. In general, we have

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

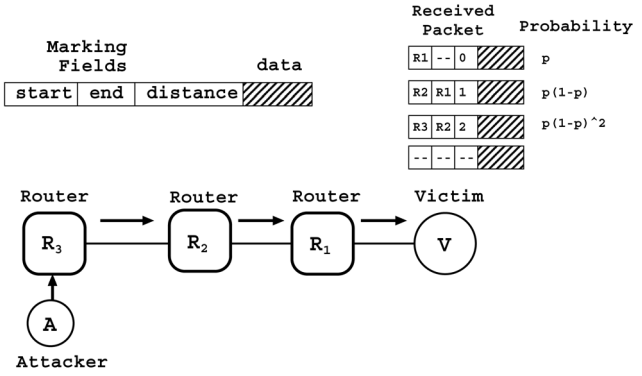


Fig. 2. Example of a probabilistic edge marking.

In other words, an edge which is d hops away from the victim site $\mathcal{V}$ will only be marked if a router connected to that edge decides to mark the packet and the remaining routers along the same path decide not to mark this packet (thereby overwriting any old mark). Let $P_u(d)$ be the probability that a victim site $\mathcal{V}$ will *not* find an edge which is d hops away or closer as a marked edge. We have

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

This happens when all the routers along the path to $\mathcal{V}$ decide not to mark the packet. Fig. 2 illustrates the set of marked and unmarked edges collected by the victim site $\mathcal{V}$ under a simple linear network topology. In the example, $\mathcal{V}$ can collect four types of packets. The first three types are marked packets with edges (R_3, R_2), (R_2, R_1), and $(R_1, -)$, respectively. The last type of packet that can be received by $\mathcal{V}$ is the unmarked packet.

$\mathcal{V}$, upon receiving packets, needs to first filter out those unmarked packets since they do not carry any information useful in the attack graph construction. For all the collected marked packets, the victim site $\mathcal{V}$ needs to execute the *graph construction* algorithm, shown in Fig. 3, to construct the attack graph.

To illustrate the attack graph construction algorithm, let us consider a network which has a tree-like topology as depicted in Fig. 4. In the figure, the routers are represented

Algorithm: Attack Graph Construction Procedure at Victim site $\mathcal{V}$:

```
Initialize the tree G to have a root node which
                    is the victim site V;
filter out unmarked packet;
sort marked packet in ascending order of the value
                    of the distance field;
/* attach each marked edge to the tree G */
for (each marked packet w) {
  if (w.distance == 0) {
    insert edge (w.start,V,0) into G;
  }
  else {
    if ((w.end == one of the outermost node in G) and
        (w.distance == that outermost node's distance))
      insert edge (w.start,w.end,w.distance) into G;
  }
}
extract path (Ri..Rj) by enumerating acyclic paths in G;
```

Fig. 3. Attack graph construction algorithm.

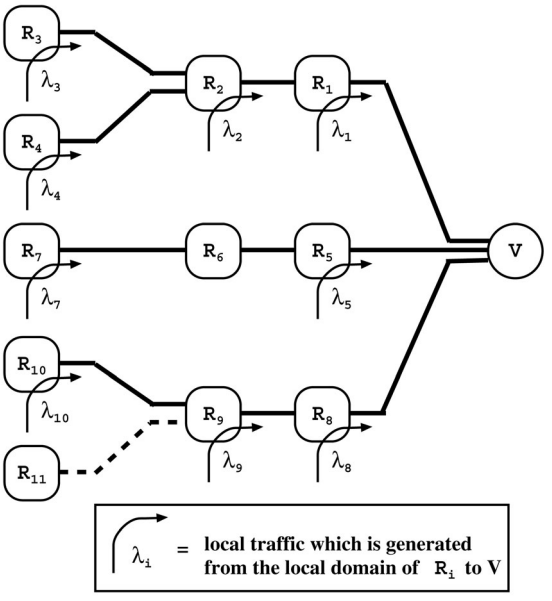


Fig. 4. Example of attack graph construction: Solid lines represent edges of the constructed attack graph $\mathcal{G}$ while a dotted line represents an edge that is not marked or discovered by the algorithm.

by R_i and the victim site is represented by $\mathcal{V}$. Packets passing through the routers will be marked by the probabilistic edge marking algorithm shown in Fig. 1. At the end of the measurement period, the victim site $\mathcal{V}$ will have received a number of packets with the marking classification shown in Table 1.

$\mathcal{V}$ uses these marked packets to create an attack graph based on the attack graph construction algorithm in Fig. 3. From the above example, the attack graph contains four linear paths, which are:

1. **Path 1:** $R_3 \rightarrow R_2 \rightarrow R_1 \rightarrow \mathcal{V}$,
2. **Path 2:** $R_4 \rightarrow R_2 \rightarrow R_1 \rightarrow \mathcal{V}$,
3. **Path 3:** $R_7 \rightarrow R_6 \rightarrow R_5 \rightarrow \mathcal{V}$, and
4. **Path 4:** $R_{10} \rightarrow R_9 \rightarrow R_8 \rightarrow \mathcal{V}$.

It is important to point out that the following advantages of the probabilistic marking algorithm:

- One only needs to allocate a small and finite amount of space in the IP header so as to store the edge information. As proposed in [14], this can be achieved using the existing unused space in the header.
- The distance field provides an *ordering relationship* between edges so that the victim site $\mathcal{V}$ can construct the attack graph from the received marked edges.

TABLE 1
Marked Packet Received by the Victim Site $\mathcal{V}$

Distance from $\mathcal{V}$	Packet Marking Type
–	un-marked packet
0	$(R_1,--,0)$, $(R_5,--,0)$, $(R_8,--,0)$
1	$(R_2, R_1,1)$, $(R_6, R_5,1)$, $(R_9, R_8,1)$
2	$(R_3, R_2,2)$, $(R_4, R_2,2)$, $(R_7, R_6,2)$, $(R_{10}, R_9,2)$

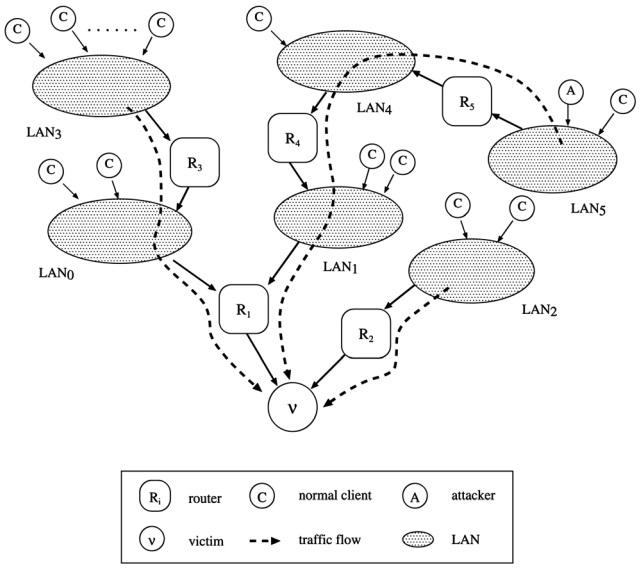


Fig. 5. An example network topology.

- One can use this algorithm to construct the attack graph in real time (e.g., construction of the attack graph can be done during a DDoS attack).

One major shortcoming of the probabilistic edge marking algorithm is that it does not provide an effective means to *locate* the positions of the potential attackers in real time. In general, the attack graph only provides a topology of the attack traffic addressed to the victim site $\mathcal{V}$. In the following, we present a methodology to identify the potential attackers and, at the same time, quantify the minimum time required to determine the locations of the attackers.

3 Attacker Traceback: Linear Topology

Let us first present a network model and some of its important components, namely *transit traffic*, *local traffic*, and *outgoing traffic* of a participating router. Fig. 5 illustrates the network model. A directed acyclic graph (DAG) rooted at $\mathcal{V}$ represents the network topology while $\mathcal{V}$ itself also represents a victim site. The DAG consists of routers and local area networks (LANs). For simplicity of illustration, the DAG shows only the network components that participate in forwarding traffic to the victim site $\mathcal{V}$. Let R_i represent an upstream router of $\mathcal{V}$ and the DAG a map of all routers which forward traffic to $\mathcal{V}$.

LANs contain a number of end hosts, including some legitimate clients of $\mathcal{V}$ and possibly some attackers. The traffic to $\mathcal{V}$ generated by both the clients and attackers is forwarded by routers of their LANs. For example, in Fig. 5, router R_1 serves as a gateway between LAN_0 and LAN_1. These two LANs are regarded as the *local administrative domain* of R_1. A router is responsible for forwarding traffic generated from its local administrative domain and any traffic generated from the local administrative domains of its *upstream routers*. In Fig. 5, the routers R_3, R_4, and R_5 are considered the upstream routers of R_1 while routers R_3 and R_4 are the *immediate upstream routers* of R_1. We say that a router is a *leaf* router in a DAG if it is not connected to any upstream router. For example, in Fig. 5, routers R_2, R_3, and R_5 are leaf routers. All other routers are called *internal* routers, for example, R_1 and R_4.

In our work, we classify three types of traffic. They are *transit traffic*, *local traffic*, and *outgoing traffic*. The transit traffic of R_i is the traffic forwarded from the immediate upstream routers of R_i. The local traffic of R_i is the traffic generated from the local administrative domain of R_i. The outgoing traffic of R_i is the sum of the transit traffic and the local traffic of R_i. To illustrate, consider the example in Fig. 5. The traffic to $\mathcal{V}$ is generated in LAN_5 and has to go through routers R_5, R_4, and R_1. The traffic from LAN_5 is considered the transit traffic for router R_4. Clients in LAN_4 can also generate traffic to $\mathcal{V}$ and this traffic is considered the local traffic of R_4. The union of these two kinds of traffic generated in LAN_4 and LAN_5 is considered the outgoing traffic of R_4.

Definition 1. *The "local traffic" of a router R_i consists of the packets destined for the victim site $\mathcal{V}$, where the packets are generated within the local administrative domain of R_i.*

Definition 2. *An attacker is defined as the router whose local traffic contributes a sufficiently large percentage of the received traffic at the victim site $\mathcal{V}$.*

In our work, we propose a traceback algorithm to determine the locations of the attackers. Intuitively, attackers may send a large number of packets to the victim site $\mathcal{V}$. Therefore, the volume of local traffic of routers where attackers are located must be high. If one can determine the amount of the local traffic generated from the local administrative domain of each router, one can traceback the source of the attack up to that local administrative domain.

Note that the victim site $\mathcal{V}$ can utilize the marked packets in two ways. First, these marked packets are used to construct an attack graph $\mathcal{G}$. Second, one can also use the marked packets to generate *statistical information* so as to traceback the potential attackers. In the following, we will show that, based on the collected marked packets, one can deduce the intensity of the *local traffic* for every router in the attack graph $\mathcal{G}$. Based on the traffic intensity of the local traffic rate at each router, one can determine the possible locations of the attackers. It is important to point out that one major technical difficulty in estimating the local traffic rate of a router is to determine the *minimum stable time* t_{min}, which is the minimum time we need to collect the marked packets such that the calculated local traffic rates are *correct* and *stable*. Notice that once we reach the stability point, then we can accurately determine the locations of the attackers. In the following two sections, we illustrate how we can determine the local traffic rates and the minimum stable time t_{min}.

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

To illustrate the proposed method, let us consider the following simple but illustrative example (in a later section, we will extend the method to a general network topology). Fig. 6 illustrates a linear network topology with d routers R_i, $1 \leq i \leq d$. Router R_i receives its local traffic (i.e., traffic generated from its local network domain) and forwards the

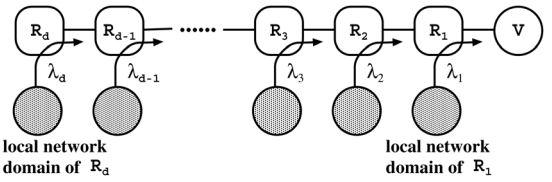


Fig. 6. Linear topology with local traffic rate λ_i to the victim site $\mathcal{V}$.

local traffic to the victim site $\mathcal{V}$. Let λ_i denote the average local traffic rate (in packets/s) from the router R_i to the victim site $\mathcal{V}$. Each router also performs the packet routing/forwarding functions for all upstream traffic destined for $\mathcal{V}$. For example, router R_3 in Fig. 6 performs packet routing/forwarding for traffic λ_j for $j = 3, 4, \ldots, d$. The purpose 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 intensities of these traffic rates, one can identify the locations of the attackers. Note also that in estimating the local traffic rates λ_i, one cannot simply rely on inspecting the source IP addresses since they can be easily spoofed by the attackers.

By the nature of the probabilistic marking algorithm, each router may mark any local or upstream transit packet to the victim site $\mathcal{V}$. If the router R_j decides to mark a transit packet, then, in essence, it resets the packet's marking by any upstream router of R_j. Let $\tilde{N}_j(t)$ be a random variable denoting the number of marked packets received by the victim site $\mathcal{V}$ at time t, such that the `start` field is the router R_j, for $1 \le j \le d$. To compare these random variables $\tilde{N}_j(t)$ for $1 \le j \le d$, we need the following definition from [13]:

Definition 3. *Let X_1 and X_2 be two random variables. We say that X_1 is stochastically larger than X_2, denoted as $X_1 \ge_{st} X_2$, if*

$$\text{Prob}[X_1 > x] \ge \text{Prob}[X_2 > x] \quad \forall x.$$

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

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

The above relationship holds because

1. Each router carries its local traffic as well as any upstream transit traffic to victim site $\mathcal{V}$. In the long run, the number of *marked* packets by R_j should be 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 site $\mathcal{V}$. Therefore, the router R_j erases any edge marking by an upstream router.

Let $N_j(t)$ be the number of marked packets received by the victim site $\mathcal{V}$ at time t such that the `start` field is the router R_j. We have the following relationship:

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

The term in the parenthesis corresponds to the total number of packets destined for the victim site $\mathcal{V}$ forwarded by router R_j 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, so we can rearrange the above equations and obtain the local traffic rate λ_i at each router R_i as:

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

The remaining technical issue is the following: To have an accurate estimate of the local traffic rates, we have to make sure that the conditions in (3) are satisfied. It is not difficult to observe that these conditions are not satisfied, for example, when the duration of collecting these marked packets 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 (3) are satisfied. Once these conditions are satisfied, we can use the estimate of the local traffic rates based on (5) to traceback the potential attackers. To simplify the notation, let us define:

$$N_j(t) = \lambda'_j t, \text{ where } \lambda'_j = \left(\sum_{i=j}^{d} \lambda_i\right) p(1-p)^{j-1}. \tag{6}$$

Assuming that the random variables $\tilde{N}_j$ and $\tilde{N}_{j+1}$ are Poisson random variables,[1] we can reformulate the problem of finding the minimum stable time t_{min} as:

$$\text{Prob}[\tilde{N}_j(t_{min}) \ge \tilde{N}_{j+1}(t_{min})] \ge p_{threshold} \quad \forall j \in \{1, \ldots, d-1\}, \tag{7}$$

where $p_{threshold}$ is a high probability (e.g., $p_{threshold} = 95\%$). In other words, we want to find the minimum time t_{min} such that, with a very high probability, the number of collected packets marked by router R_j is higher than the number of collected packets marked by router R_{j+1} for $1 \le j \le d-1$. Since the random variables $\tilde{N}_j(t)$ and $\tilde{N}_{j+1}(t)$ are Poisson, we have

$$\begin{aligned} &\text{Prob}[\tilde{N}_j(t_{min}) \ge \tilde{N}_{j+1}(t_{min})] \\ &= \sum_{k=0}^{\infty} \text{Prob}[\tilde{N}_j(t_{min}) \ge k] \text{Prob}[\tilde{N}_{j+1}(t_{min}) = k] \\ &= \sum_{k=0}^{\infty} \left[\sum_{n=k}^{\infty} \frac{(\lambda'_j t_{min})^n}{n!} e^{(-\lambda'_j t_{min})}\right] \frac{(\lambda'_{j+1} t_{min})^k}{k!} e^{(-\lambda'_{j+1} t_{min})} \\ &\qquad \forall j \in \{1, \ldots, d-1\}. \end{aligned} \tag{8}$$

Based on the above expression, one can easily determine the minimum stable time t_{min} using standard numerical methods [5] with a worst cast complexity of $O(n^2)$ [9]. Fig. 7 illustrates a *local traffic estimation procedure* to estimate the local traffic rate of each router in an attack graph $\mathcal{G}$.

3.2 Elimination of Attackers

In the previous section, we have presented a method for estimating the local traffic rate for each router in the attack graph $\mathcal{G}$. In this section, we present the algorithm to find the potential attackers and eliminate their attack traffic.

1. In a later section, we will illustrate that even if the arrival process is *non-Poisson*, the estimate is still quite robust for evaluating t_{min} and the intensity of the local traffic.

Algorithm: Local Traffic Estimation Procedure at victim site $\mathcal{V}$:

let $\mathcal{P}$ be a linear path to the victim site $\mathcal{V}$;
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 Eq. (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 rate λ_i for router R_i in $\mathcal{S}_{\mathcal{P}}$.

Fig. 7. Algorithm to determine local traffic rate for every router in the attack graph.

Given a linear path $\mathcal{P}$ in the attack graph, the victim site $\mathcal{V}$ can choose to reduce its traffic loading by a predetermined fraction $\mathcal{T}_{cutoff}(\mathcal{P})$. To reduce the traffic, the victim site $\mathcal{V}$ determines the local traffic rates for all routers in the path $\mathcal{P}$, which are sorted in nonincreasing order. The victim site $\mathcal{V}$ then signals a subset of the routers along the path $\mathcal{P}$ and instructs these routers to stop sending their local traffic to $\mathcal{V}$. Notice that a router can refuse packet forwarding not by examining the source IP address (which can be spoofed), but rather by examining the packet's data link layer information—for example, the link address of the attached LAN. Fig. 8 illustrates the procedure to eliminate the attacker's traffic to victim site $\mathcal{V}$.

4 Attacker Traceback: General Topology

In the previous section, we have presented a method to deduce the *local traffic rate* of each router as well as the *minimum stable time* t_{min} such that we can determine the locations of the potential attackers in a linear network topology. In this section, we extend the method to a general attack graph topology. The basic ideas are similar to those

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

Let $\mathcal{P}$ be a linear path to the victim site $\mathcal{V}$;
Let $\mathcal{S}_{\mathcal{P}}$ be the set of routers along the path $\mathcal{P}$;
Derive the local traffic rate 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 site $\mathcal{V}$;

Fig. 8. Algorithm to find potential attackers and eliminate their attack traffic to $\mathcal{V}$.

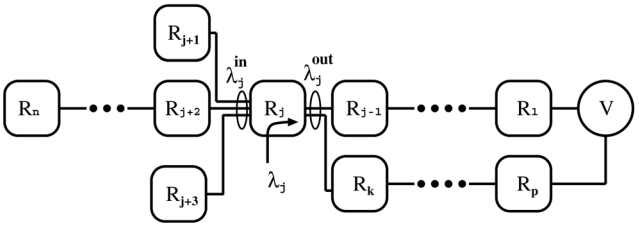


Fig. 9. A general attack graph $\mathcal{G}$.

presented in the previous section. However, we will need to refine some conditions for the general topology setting.

Fig. 9 illustrates a general topology in which each router may be connected by several upstream and downstream routers. For example, the router R_j in the figure has three upstream routers R_{j+1}, R_{j+2}, and R_{j+3} and two downstream routers R_{j-1} and R_k. For this attack graph, the farthest router from the victim site $\mathcal{V}$ is router R_n, which is of distance $d \geq 1$ hops away from $\mathcal{V}$. Each router receives its local traffic from its own network domain. Some of this traffic is destined for the victim site $\mathcal{V}$. Let λ_j denote the average local traffic rate (in pkts/s) from the router R_j to the victim site $\mathcal{V}$. Let λ_j^{in} denote the total average traffic rate from all upstream routers of R_j to the victim site $\mathcal{V}$ and λ_j^{out} denote the total average traffic rate from router R_j to all its downstream routers addressed to the victim site $\mathcal{V}$. Again, the goal is to deduce the *local traffic rate*λ_j for all routers in an attack graph. Then, based on their local traffic intensities, we can identify the locations of the potential attackers.

Let $\mathcal{G}$ denote the attack graph based on the attack graph construction method described in Section 2. We say that a router is a *leaf* router in the attack graph $\mathcal{G}$ if it is not connected to any upstream router. For example, in Fig. 9, router R_n is a leaf router (and it is an upstream router of routers R_{j+1} to R_{j+3}). All other routers are called *internal* routers. Let (i, j) denote an edge between router R_i and R_j. We define $\lambda_{(i \to j)}$ to be the average traffic rate to the victim site $\mathcal{V}$ that passes through the edge (i, j) between router R_i and R_j.

The average local traffic rate λ_j for router $R_j \in \mathcal{G}$ is:

$$\lambda_j = \begin{cases} \lambda_j^{out} & \text{if } R_j \text{ is a leaf router,} \\ \lambda_j^{out} - \lambda_j^{in} & \text{if } R_j \text{ is an internal router,} \end{cases} \tag{9}$$

where λ_j^{in} and λ_j^{out} are the average traffic rates into and out of router R_j, respectively. These average traffic rates can be computed by

$$\lambda_j^{in} = \sum_{\substack{\forall (i,j) \text{ where } R_i \text{ is an} \\ \text{upstream router of } R_j}} \lambda_{(i \to j)}, \tag{10}$$

$$\lambda_j^{out} = \sum_{\substack{\forall (j,k) \text{ where } R_k \text{ is a} \\ \text{downstream router of } R_j}} \lambda_{(j \to k)}. \tag{11}$$

Therefore, if we can estimate the average traffic rate $\lambda_{(i \to j)}$ for all marked edges (i, j) in the attack graph $\mathcal{G}$, we can deduce the average local traffic rate of each router based on (9) to (11). In Fig. 10, we present the procedure for estimating $\lambda_{(i \to j)}$ for each marked edge (i, j) in $\mathcal{G}$.

Algorithm: Estimating the Average Traffic Rate $\lambda_{(i\to j)}$ for each edge (i,j) in $\mathcal{G}$

Let $\mathcal{G}$ be an attack graph with root $\mathcal{V}$;
Each marked edge (i,j) in $\mathcal{G}$ has $(R_i, R_j, d_{(i\to j)}, N_{(i\to j)})$
/* R_i = start address of the marked edge (i,j)
R_j = end address of the marked edge (i,j)
$d_{(i\to j)}$ = hop count between R_i and victim $\mathcal{V}$
$N_{(i\to j)}$ = the number of times this marked edge (i,j) has been collected */
Set $\lambda_{(i\to j)} = 0$ for all edges (i,j) in $\mathcal{G}$;
For each edge in $\mathcal{G}$ {
/* go through each edge in $\mathcal{G}$ */
add $\lambda_{(i\to j)}$ by $\frac{N_{(i\to j)}}{tp(1-p)^{d_{(i\to j)}}}$
}
output: average traffic rate $\lambda_{(i\to j)}$ for each edge (i,j) in $\mathcal{G}$.

Fig. 10. Procedure to estimate $\lambda_{(i\to j)}$ for every marked edge (i,j) in $\mathcal{G}$.

Let $\tilde{N}_j^{out}(t)$ be the random variable denoting the number of marked packets received by the victim site $\mathcal{V}$ at time t such that the `start` field is the router R_j. Let $\tilde{N}_j^{in}(t)$ be the random variable denoting the number of marked packets received by the victim site $\mathcal{V}$ at time t such that the `end` field is the router R_j. After a sufficient amount of time in collecting these marked packets, we have the following relationship:

$$\tilde{N}_j^{out}(t) \geq_{st} \tilde{N}_j^{in}(t) \qquad \forall R_j \in \mathcal{G}. \tag{12}$$

The above relationship holds because

1. There is nonnegative local traffic originated from router R_j to the victim site $\mathcal{V}$. Therefore, the number of *marked* output packets from R_j should be greater than the number of *marked* input packets to R_j in the long run.
2. The probabilistic edge marking algorithm will mark any transit packet to $\mathcal{V}$ from upstream of router R_j. Therefore, the router R_j may erase any edge marking of a transit packet from its upstream routers.

The remaining issue is: To have an accurate estimate of the local traffic rate λ_j, we have to guarantee that the conditions in (12) are satisfied. Once the conditions of (12) are satisfied, we can then estimate the local traffic rate λ_j based on (9), (10), and (11) to traceback the potential attackers. In the following, we provide an analytical method to derive the minimum stable time t_{min} such that the conditions of (12) are satisfied.

Let $N_j^{out}(t)$ be the number of marked packets received by the victim site $\mathcal{V}$ at time t such that the `start` field is the router R_j. Let $N_j^{in}(t)$ be the number of marked packets received by the victim site $\mathcal{V}$ at time t such that the `end` field is equal to router R_j. We have the following relationship:

$$N_j^{in}(t) = \sum_{\forall(i,j)} \lambda_{(i\to j)}(t) p(1-p)^{d_{(i\to j)}-1} t, \tag{13}$$

$$N_j^{out}(t) = \sum_{\forall(j,k)} \lambda_{(j\to k)}(t) p(1-p)^{d_{(j\to k)}-1} t, \tag{14}$$

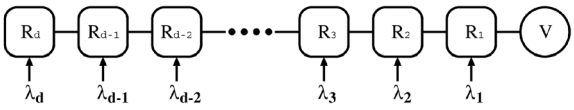


Fig. 11. Network topology for experiments 1 and 2.

where $\lambda_{(i\to j)}(t)$ and $\lambda_{(j\to k)}(t)$ are the traffic rate estimates of the corresponding edges (i,j) and (j,k) at time t according to the procedure given in Fig. 10, and $d_{(i\to j)}$ and $d_{(j\to k)}$ are the hop count from router R_i to victim $\mathcal{V}$ and from router R_j to victim $\mathcal{V}$, respectively.

To simplify the notation, let us define $\lambda'^{in}_j(t)$ and $\lambda'^{out}_j(t)$ as:

$$\begin{aligned} \lambda'^{in}_j(t) &= \sum_{\forall(i,j)} \lambda_{(i\to j)}(t) p(1-p)^{d_{(i\to j)}-1}, \\ \lambda'^{out}_j(t) &= \sum_{\forall(j,k)} \lambda_{(j\to k)}(t) p(1-p)^{d_{(j\to k)}-1}, \\ N_j^{in}(t) &= \lambda'^{in}_j(t)\, t \text{ and } N_j^{out}(t) = \lambda'^{out}_j(t)\, t. \end{aligned} \tag{15}$$

We can reformulate the problem of finding the minimum stable time t_{min} such that:

$$\text{Prob}[\tilde{N}_j^{out}(t_{min}) \geq \tilde{N}_j^{in}(t_{min})] \geq p_{threshold} \quad \forall R_j \in \mathcal{G}, \tag{16}$$

where $p_{threshold}$ is a high probability (e.g., $p_{threshold} = 95\%$). The objective is: Find the minimum time such that with a very high probability, the number of collected packets marked by the router R_j is higher than the number of collected packets marked by the upstream routers of R_j, for all routers in the attack graph $\mathcal{G}$. Assuming that the random variable is Poisson, we have

$$\begin{aligned} &\text{Prob}[\tilde{N}_j^{out}(t_{min}) \geq \tilde{N}_j^{in}(t_{min})] \\ &= \sum_{k=0}^{\infty} \text{Prob}[\tilde{N}_j^{out}(t_{min}) \geq k] \text{Prob}[\tilde{N}_j^{in}(t_{min}) = k] \\ &= \sum_{k=0}^{\infty} \left[\sum_{n=k}^{\infty} \frac{\left[\lambda'^{out}_j(t_{min}) t_{min}\right]^n}{n!} e^{-\left[\lambda'^{out}_j(t_{min}) t_{min}\right]} \right] \times \\ &\quad \frac{\left[\lambda'^{in}_j(t_{min}) t_{min}\right]^k}{k!} e^{-\left[\lambda'^{in}_j(t_{min}) t_{min}\right]} \quad \forall R_j \in \mathcal{G}. \end{aligned} \tag{17}$$

Again, we can easily determine the minimum stable time t_{min} using standard numerical methods [5].

5 Experiments

In this section, we perform experiments to illustrate the effectiveness in locating the attackers. In particular, we show the correctness and robustness of using (8) and (17) to find the minimum stable time t_{min} such that we can compute the intensities of the local traffic rates for all the routers in the attack graph $\mathcal{G}$. Note that a smaller value of t_{min} implies that we can find the locations of the attackers earlier. We illustrate various factors which can influence the value of the minimum stable time t_{min}. We also extend the analysis of the attacker location method to a general network topology. In the first two sets of experiments, we use a linear network topology as shown in Fig. 11. To derive the attack graph, the victim $\mathcal{V}$ requests the participating routers to mark packets

TABLE 2
Minimum Stable Time: Theoretical versus Simulation Results for Different Packet Arrival 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

with probability $p = 1/25$. To estimate whether we have reached the stability conditions specified by (8) or (17), we set $p_{threshold} = 95\%$. In other words, the estimates of the local traffic rates should be highly accurate. We define a variable, the **attack traffic ratio** (ATR), which is the ratio of the attacker's average traffic rate to the normal average local traffic rate. For example, if the normal average local traffic rate is 100 pkts/s and ATR is 20, then the attacker's average traffic rate is equal to 2,000 pkts/s.

EXPERIMENT 1: (Correctness and robustness of estimating the minimum stable time t_{min}). This set of experiments evaluates the correctness of the estimate of the minimum stable time t_{min} based on the mathematical model (8) presented in Section 3.1. We show that the estimate is robust and accurate for different traffic arrival processes, including Poisson arrivals, constant-rate arrivals, burst mode arrivals, and packet arrivals generated under a random on/off process modulated by a Markov process [11]. We demonstrate the performance trade-off with different values of the parameter $p_{threshold}$. For this set of experiments, we use a normal average local traffic rate of 100 pkts/s. The ATR is set to 20 (in other words, the attacker's average traffic rate is 2,000 pkts/s).

Experiment 1.A (*Influence on the minimum stable time t_{min} by different packet arrival processes*). In this experiment, we use a linear topology in Fig. 11 with 25 routers (e.g., $d = 25$). There is one attacker located at the farthest router R_{25}. The normal traffic and the attack traffic are generated using three methods: 1) Poisson process, 2) Constant rate (e.g., an average rate of 100 pkts/s implies that every 0.01 second, there is a new packet generated by a router), and 3) Burst rate (e.g., an average rate of 100 pkts/s 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 (8). Table 2 illustrates that, with different values of t_{min}, our computed $p_{threshold}$ value is very close to the simulated $p_{threshold}$ values for different traffic generation processes. This indicates that our methodology is quite accurate and robust in estimating the minimum stable time t_{min}. In addition, regardless of the packet arrival process, it takes around 25 seconds to determine the local traffic rates of all the routers in the attack graph $\mathcal{G}$.

Experiment 1.B (*Influence on the minimum stable time t_{min} by different packet arrival processes under random on/off phase*). We also consider other ways to generate the normal and attack traffic. In particular, we consider a random on-off process

TABLE 3
Minimum Stable Time: Theoretical versus Simulation Results for Different Packet Arrival Processes under Random On/Off Phase with $\lambda = 2.0$ and $\mu = 1.0$

t_{min} (sec)	$p_{threshold}$ (%)			
	Theoretical Computed Values	random on/off Pkt Generation $\lambda = 2, \mu = 1$		
		Poisson	Constant	Burst
5.0	79.46	79.70	79.70	79.50
10.0	87.35	88.40	87.90	87.90
15.0	91.80	91.50	92.00	90.80
20.0	94.55	94.40	94.90	94.20
25.0	96.31	95.50	96.20	96.60

modulated by a Markov process $\{X(t)\}$, where $X(t) \in \{0, 1\}$. Let λ (μ) be the transition rate from state 0 (respectively, state 1) to state 1 (respectively, state 0). If the current state of the Markov process is 1, then packet generation is enabled. If the state of the process is 0, then no packet is generated into the system. Again, the packet generation at state 1 can be classified into three types, namely, 1) Poisson process with an average rate of 100 pkts/s, 2) Constant rate (e.g., an average rate of 100 pkts/s implies that every 0.01 second, there is a new packet generated by a router), and 3) Burst rate (e.g., an average rate of 100 pkts/s implies that we generate 100 packets in one burst for every second). We consider two different sets of parameters, 1) $\lambda = 2.0, \mu = 1.0$ and 2) $\lambda = 10, \mu = 30$. Tables 3 and 4 illustrate the influence on the minimum stable time t_{min} under these two settings. Again, we observe that the estimate is quite accurate and robust in estimating the minimum stable time t_{min} even when the packet arrival process is non-Poisson. Another important observation is that we can quickly estimate t_{min}; in particular, we need around 25 seconds to reach the 95 percent stability condition. This implies that we can quickly determine the locations of the attackers.

Experiment 1.C (*Influence on t_{min} and variance of traffic rate estimation by different $p_{threshold}$*). Equation (8) indicates that if the stochastic relationship in (3) is satisfied with a high probability $p_{threshold}$, the estimate of local traffic will be accurate. The accuracy can be expressed by the variance of the local traffic rate estimates. In this experiment, we illustrate how the parameter $p_{threshold}$ affects the minimum

TABLE 4
Minimum Stable Time: Theoretical versus Simulation Results for Different Packet Arrival Processes under Random On/Off Phase with $\lambda = 10$ and $\mu = 30$

t_{min} (sec)	$p_{threshold}$ (%)			
	Theoretical Computed Values	random on/off Pkt Generation $\lambda = 10, \mu = 30$		
		Poisson	Constant	Burst
5.0	79.46	81.50	77.50	78.30
10.0	87.35	88.00	86.80	86.20
15.0	91.80	91.70	92.70	90.50
20.0	94.55	93.40	93.90	94.60
25.0	96.31	97.30	95.90	96.60

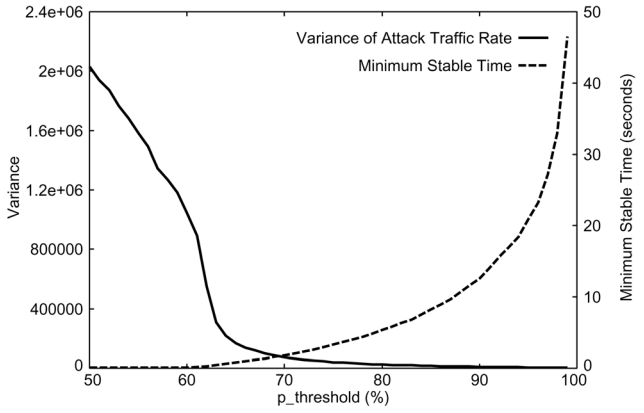


Fig. 12. Influence on t_{min} and variance of the attacker's traffic rate estimation by different $p_{threshold}$.

stable time t_{min} and the quality of the attacker's traffic rate estimate. Fig. 12 illustrates that, for different values of $p_{threshold}$, we have different values of t_{min}. The corresponding variances of the attacker's traffic rate estimates are also shown. The figure shows that the variance in the estimate of the attacker's traffic rate approaches zero, which indicates that a highly accurate estimate of attacker's traffic can be achieved when $p_{threshold} \geq 80\%$. Our experiment uses a linear topology with 25 routers and the attacker is located at the farthest router R_{25}. The normal average local traffic is 100 pkts/s and the attacker's average traffic rate is 2,000 pkts/s. The packets are generated by a Poisson process and are marked by each router with probability $p = 1/25$. First we vary $p_{threshold}$ from 50 to 99 percent and calculate the corresponding minimum stable time t_{min} by (8). Then, for each minimum stable time t_{min}, we conduct experiments to estimate the traffic rate of the attacker. We repeat the entire process 10,000 times. For each minimum stable time t_{min}, we obtain 10,000 estimated traffic rates of the attacker. Let X be the random variable denoting the estimated traffic rate of the attacker. The variance of the attacker traffic rate estimates is a measure of the spread of a distribution about its mean and is defined by $Var(X) = E([X - E(X)]^2)$. As expected, a higher value of $p_{threshold}$ gives a larger value of t_{min}, meaning that more time is needed to determine the locations of the attackers. However, since t_{min} is less than 60 seconds in all the cases, we can still locate the attackers in a pretty short time.

Experiment 1.D (*Influence on the minimum stable time t_{min} by different packet arrival processes under a three-state Markov process*). In this experiment, we perform a more elaborate performance study by generating normal and attack traffic under a three-state Markov process $\{X(t)\}$, where $X(t) \in \{0, 1, 2\}$. Let λ_0 (μ_0) be the transition rate from state 0 (state 1) to state 1 (state 0) and λ_1 (μ_1) be the transition rate from state 1 (state 2) to state 2 (state 2). Fig. 13 illustrates the

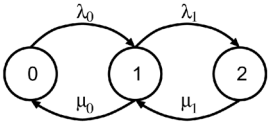


Fig. 13. Transition rates of the three-state Markov process.

TABLE 5
Minimum Stable Time: Theoretical versus Simulation Results for Different Packet Arrival Processes under the Three-State Markov Process with $\lambda_0 = 4.0$, $\lambda_1 = 3.0$, $\mu_0 = 2.0$, $\mu_1 = 1.0$

t_{min} (sec)	$p_{threshold}$ (%)			
	Theoretical Computed Values	3-State Markov Process $\lambda_0 = 4, \lambda_1 = 3, \mu_0 = 2, \mu_1 = 1$		
		Poisson	Constant	Burst
5.0	79.46	80.30	80.50	80.20
10.0	87.35	86.40	86.70	87.10
15.0	91.80	91.90	91.60	91.10
20.0	94.55	93.30	93.80	93.60
25.0	96.31	96.60	96.10	96.20

transition rates of the three-state Markov-modulated process. If the current state of the Markov process is 2, the packet generation rate is double the rate in state 1. If the state of the process is 0, then no packet is generated into the system. Again, the packet generation at state 1 and state 2 can be classified into three types, namely, 1) Poisson process with an average rate of 100 pkts/s, 2) Constant rate (e.g., an average rate of 100 pkts/s implies that every 0.01 second, there is a new packet generated by a router), and 3) Burst rate (e.g., an average rate of 100 pkts/s implies that we generate 100 packets in one burst for every second). We consider two different sets of parameters, 1) $\lambda_0 = 4.0$, $\lambda_1 = 3.0$, $\mu_0 = 2.0$, $\mu_1 = 1.0$ and 2) $\lambda_0 = 5$, $\lambda_1 = 10$, $\mu_0 = 15$, $\mu_1 = 20$. Tables 5 and 6 show the influence on the minimum stable time t_{min} under these two settings. We can observe that the estimate of the minimum stable time t_{min} is highly accurate regardless of the packet arrival process. Furthermore, it takes only around 25 seconds to reach the 95 percent stability condition, showing the effectiveness of our proposed method.

EXPERIMENT 2: (Factors which influence the minimum stable time t_{min}). In this set of experiments, 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}. For this set of experiments, we set the normal average local traffic rate to 100 pkts/s. The ATR is set to be 20 (i.e., the attacker's average traffic rate is 2,000 pkts/s).

TABLE 6
Minimum Stable Time: Theoretical versus Simulation Results for Different Packet Arrival Processes under the Three-State Markov Process with $\lambda_0 = 5$, $\lambda_1 = 10$, $\mu_0 = 15$, $\mu_1 = 20$

t_{min} (sec)	$p_{threshold}$ (%)			
	Theoretical Computed Values	3-State Markov Process $\lambda_0 = 5, \lambda_1 = 10, \mu_0 = 15, \mu_1 = 20$		
		Poisson	Constant	Burst
5.0	79.46	78.00	79.30	78.70
10.0	87.35	86.50	86.70	86.30
15.0	91.80	91.80	91.10	91.30
20.0	94.55	94.20	94.10	94.30
25.0	96.31	96.10	96.00	96.20

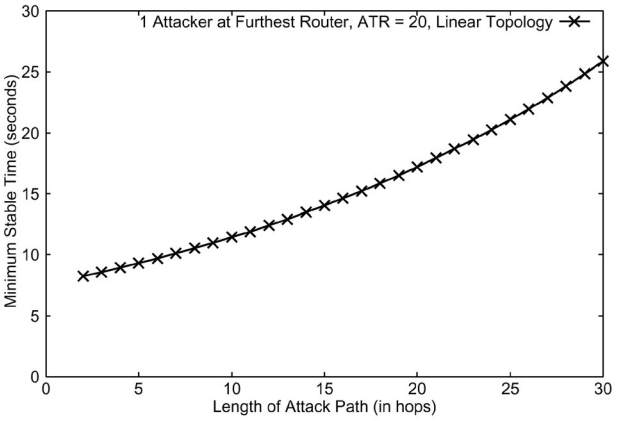


Fig. 14. Influence on t_{min} by different length of the attack path.

Experiment 2.A (*Influence on t_{min} by different lengths of the attack path*). In this experiment, the relationship between the length of an attack path and the minimum stable time t_{min} is analyzed. We use a linear network topology with varying lengths of the attack path and compute the minimum stable time t_{min}. We assume that there is one attacker located at the *farthest* router (e.g., R_d). Fig. 14 shows that as the length of the attack path increases, the minimum stable time t_{min} also increases. The reason for this linear growth is that the victim site $\mathcal{V}$ needs to take a longer time to collect a sufficient number of *marked* packets from a farther router.

Experiment 2.B (*Influence on t_{min} by the relative positions of the attackers*). In this experiment, we study how the number of attackers and their relative positions influence the value of the minimum stable time t_{min}. We consider three attackers in a linear network topology, where these attackers are distributed according to three different configurations. For configuration 1, the attackers are located at routers R_1, R_2, and R_3 (i.e., the three closest routers to the victim $\mathcal{V}$). For configuration 2, the attackers are located at the three farthest routers from the victim $\mathcal{V}$. In configuration 3, the attackers are *evenly placed* along the linear network; i.e., they are located at routers $R_{\lfloor \frac{di}{3} \rfloor}$ where d is the length of the attack path and $i = 1, 2, 3$. Fig. 15 illustrates the minimum stable time t_{min} when $\mathrm{ATR} = 20$

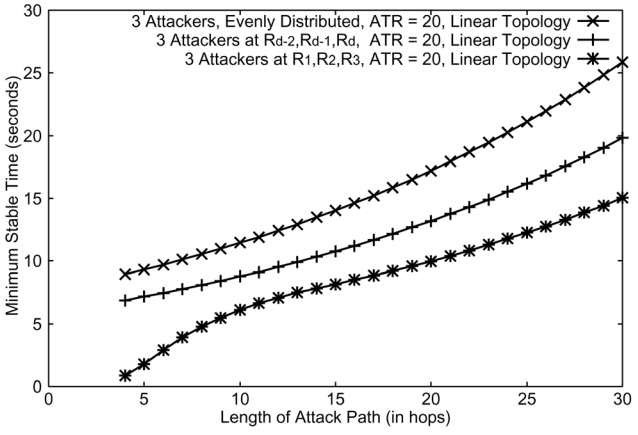


Fig. 15. Influence on t_{min} by the relative positions of the attackers.

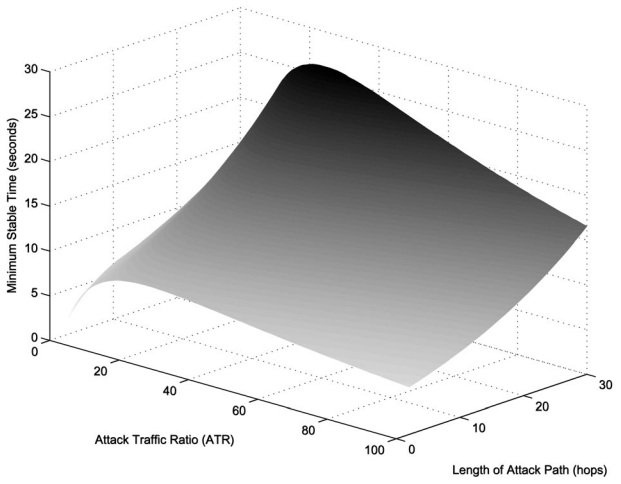


Fig. 16. Influence on t_{min} by different ATR and different length of the attack path.

and $p_{threshold} = 95\%$. We observe that when the attackers are closer to the victim $\mathcal{V}$ (i.e., configuration 1), the achieved minimum stable time t_{min} is lower than the other configurations. The minimum stable time t_{min} achieves the highest value when attackers are evenly placed in the network (i.e., configuration 3). The reason for this ordering is that the estimate of local traffic rate at the farthest router takes the longest to become stable while the estimate of the local traffic rate at the nearest router takes the least time. Regardless of the placement of the attacker, we can determine their locations within 30 seconds, illustrating the effectiveness of the proposed method.

Experiment 2.C (*Influence on t_{min} by different ATR and different lengths of the attack path*). In this experiment, we illustrate how the attack traffic ratio (ATR) and the length of the attack path can affect the value of the minimum stable time t_{min}. We consider a linear topology and compute the minimum stable time t_{min} with varying attack traffic ratios and lengths of the attack path. Since the previous result suggests that t_{min} is largest when the attacker is located at the farthest router, we consider the worst case scenario that one attacker is located at the farthest router R_d. Fig. 16 illustrates the achieved minimum stable time t_{min} under different values of ATR and lengths of the attack path. We observe that if the attack path is longer, the minimum stable time t_{min} is larger. On the other hand, when the attack traffic ratio is higher (which is usually the case for a DDoS attack), the minimum stable time t_{min} is lower. The achieved t_{min} is less than 30 seconds in all the cases. Again, this shows that we can quickly discover the locations of the potential attackers in a DDoS attack.

EXPERIMENT 3: (Extension to General Network Topology). In the previous set of experiments, we analyze the performance of the attacker location method by assuming a linear network topology. In this set of experiments, we extend the performance study to a general network topology. We evaluate the minimum stable time t_{min} of the attacker location method based on the mathematical model (17) presented in Section 4. Fig. 17 shows a general network topology of 120 routers. We assume that

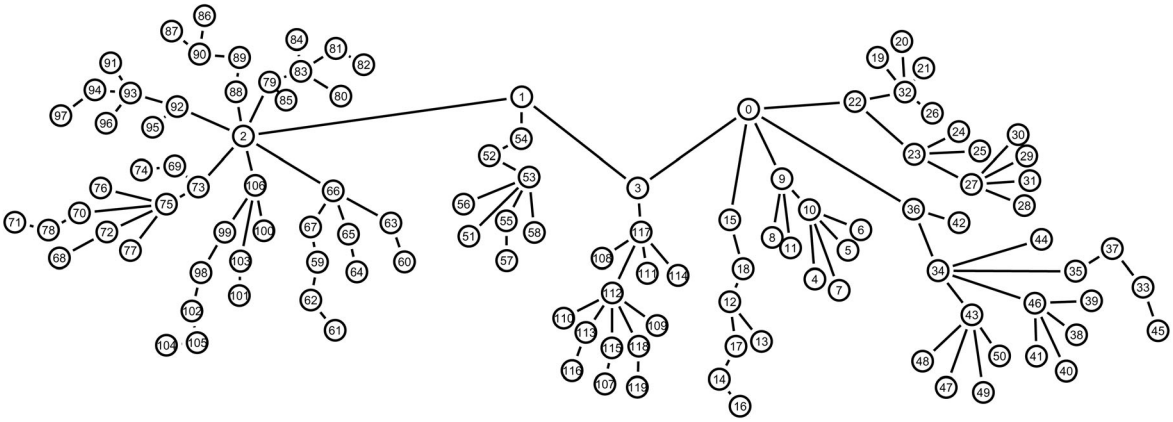


Fig. 17. General network topology with 120 routers with node 0 being the victim V.

victim $\mathcal{V}$ is located at router R_0 and the attackers are randomly assigned to different locations in disjoint attack paths. We set the normal average local traffic rate to 10 pkts/s and vary the attacker average traffic rates to be 8,000, 9,000, and 10,000 pkts/s, respectively (corresponding to ATRs of 800, 900, and 1,000).

Experiment 3.A (*Influence on t_{min} by different ATRs and different network topology diameters*). In this experiment, we vary the diameter of the general attack graph and evaluate the average minimum stable time t_{min} under different attacker traffic rates. Fig. 18 illustrates that, as the diameter of the general attack graph increases, the average minimum stable time t_{min} also increases. The reason is that it takes a longer time to collect a sufficient number of marked packets from the farthest router. Moreover, when the ATR is higher (which is a common scenario for a DDoS attack), the average minimum stable time t_{min} will be smaller. This experiment shows that we can determine the locations of the attackers within 14 seconds in the general network topology, showing the effectiveness of the proposed method.

Experiment 3.B (*Influence on t_{min} by different numbers of attackers*). In this experiment, we vary the the number of attackers in the general attack graph and evaluate the average minimum stable time t_{min} under different attacker traffic rates. The attackers are located at the farthest routers of the attack graph (note that this corresponds to a worst case study). Fig. 19 shows that as the number of attackers increases, the average minimum stable time t_{min} decreases. The reason for the decrease is that with a larger number of attackers, more marked packets will be collected. Hence, it takes a shorter time to satisfy the stability conditions of the local traffic estimation. This illustrates that we can quickly determine the locations of the attackers under a DDoS attack.

EXPERIMENT 4: (Extension to Internet Topology). In the previous set of experiments, we evaluate the performance of our attacker location method in a small size general network topology. In this set of experiments, we perform a more comprehensive performance study by using a large size Internet topology obtained from Lucent Bell Labs [1]. Fig. 20 shows an Internet map[2] generated by the traceroute data set. The testing data set contains 10,000 distinct routers and the maximum height (or the maximum hop count from the victim site $\mathcal{V}$ to the farthest router in the graph) is 23. The source of traceroute is considered the victim site $\mathcal{V}$ and the traceroute data set is used to map he upstream routers. We use the data set and repeat similar experiments as in Experiment 3. We set the normal average local traffic rate to 1 pkt/s and the attacker average traffic rate to 10,000 pkts/s (i.e., ATR = 10,000).

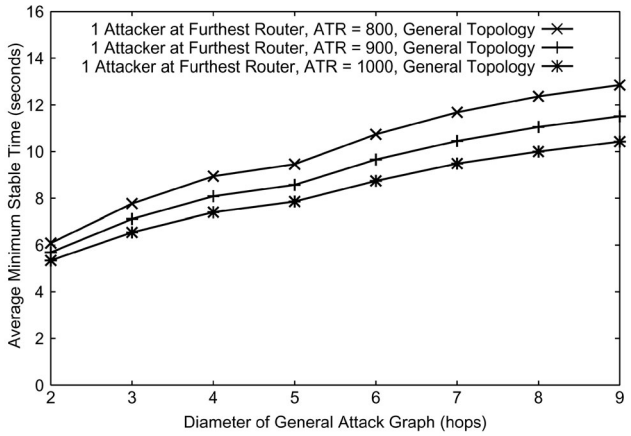


Fig. 18. Influence on t_{min} by different ATR and different diameter of the network topology.

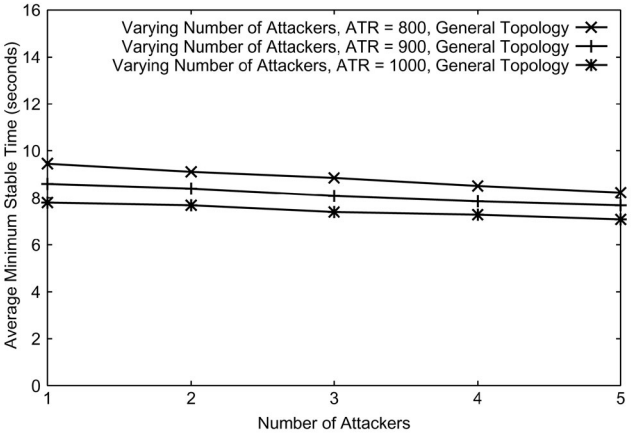


Fig. 19. Influence on t_{min} by different number of attackers.

2. Image from http://research.lumeta.com/ches/mpa Internet Mapping Project.

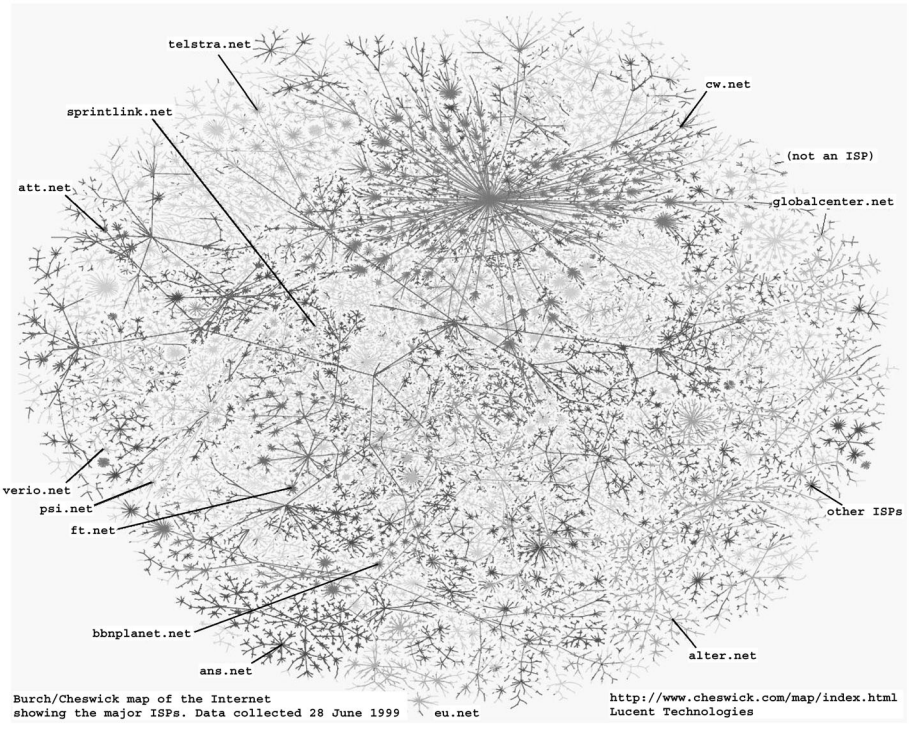


Fig. 20. Internet map generated by the traceroute data set.

Experiment 4.A (*Influence on t_{min} by different diameters of the network topology*). In this experiment, we study the relationship between the diameter of the general attack graph $\mathcal{G}$ and the average minimum stable time t_{min} in the Internet topology. The main difference between a small size topology and a large size one is that the aggregate transit traffic for routers which are closer to a victim site $\mathcal{V}$ is significantly larger than the aggregate rate for routers farther away from the victim site $\mathcal{V}$ in a large size topology. Fig. 21 illustrates the average minimum stable time t_{min} when there is a single attacker. Again, we observe that when the attacker is closer to the victim site $\mathcal{V}$, we can determine the location of the attacker earlier. One important point is that, even for this large network, it takes less than 20 seconds to determine the local traffic rate of each router and locate the attacker. This shows that our methodology is quite robust and accurate in determining the locations of the attackers.

Experiment 4.B (*Influence on t_{min} by different numbers of attackers*). In this experiment, the relationship between the number of attackers and the average minimum stable time t_{min} in the Internet topology is analyzed. We use the same traceroute data set as Experiment 4.A. The attackers are randomly assigned to different locations in disjoint attack paths. We evaluate the average minimum stable time t_{min} for different numbers of attackers. From Fig. 22, we see that the average minimum stable time t_{min} of our attacker location method decreases as the number of attackers increases. Again, this illustrates the effectiveness and

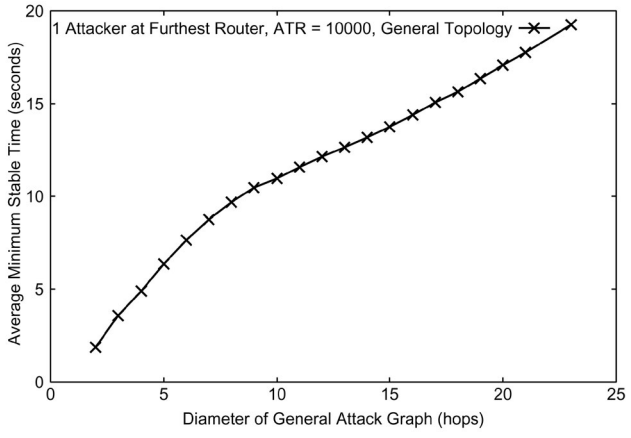


Fig. 21. Influence on t_{min} by different diameter of the network topology.

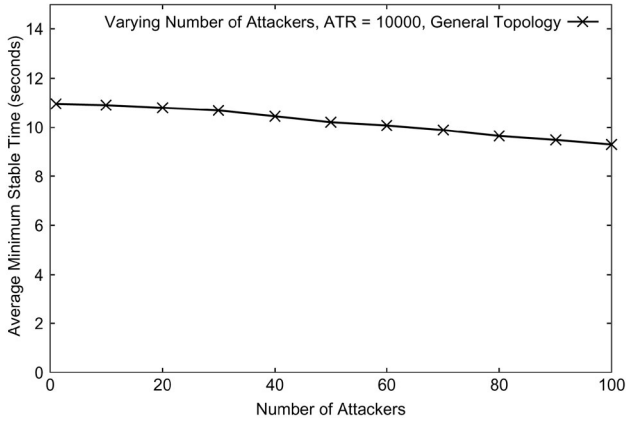


Fig. 22. Influence on t_{min} by different number of attackers.

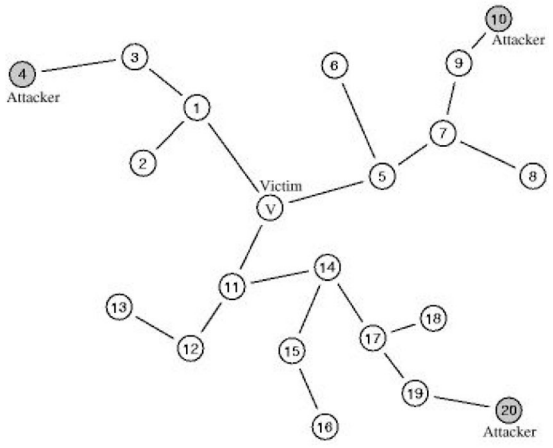


Fig. 23. Example to estimate the local traffic rates in a general topology.

robustness of our traceback method even under a large-scale DDoS attack.

Experiment 4.C (*Detect and prevent attacks mounted from multiple attackers*). To show that the proposed method can eliminate attack traffic from multiple attackers, let us consider an example in Fig. 23. We consider a general topology with 20 routers. Routers R_1 to R_3, R_5 to R_9, and R_{11} to R_{19} carry normal local traffic with a rate of 10 pkts/s to the victim site $\mathcal{V}$. There are three attackers in the network and they are located at routers R_4, R_{10}, and R_{20}, respectively. The attackers in R_4, R_{10}, and R_{20} generate attack traffic to $\mathcal{V}$ at a rate of 50 pkts/s, 75 pkts/s, and 100 pkts/s, respectively. The marking probability p for the edge marking algorithm is set to 0.1 and the stability threshold $p_{threshold}$ is set to 95 percent. In this example, the victim site $\mathcal{V}$ calculates the stability percentage every 10 seconds according to the procedure in Fig. 7. Table 7 illustrates that, after going through the loop of the traffic estimation procedure five times, we can accurately estimate the local traffic rate at each router R_i for $i = 1, \ldots, 20$. In other words, it only takes around 50 seconds to find the potential attackers in R_4, R_{10}, and R_{20}.

Table 8 illustrates the percentage of local traffic to total traffic at each router at the end of 50 seconds, the time at which the estimates of the local traffic rates become stable. As we can observe from the table, the victim site $\mathcal{V}$ can accurately estimate the intensity of the local traffic at each router in the attack graph. If the victim site $\mathcal{V}$ wishes to reduce the total traffic by 50 percent, one can set the parameter $\mathcal{T}_{cutoff}(\mathcal{P})$ to 50 percent. The victim site $\mathcal{V}$ can then signal routers R_4, R_{10}, and R_{20} to drop the attack traffic from these routers according to the procedure in Fig. 8.

TABLE 7
Estimation of the Local Traffic Rates in R_1 to R_{20}

Time (sec)	**Stability** (%)	R_4 **Traffic** (pkts/s)	R_{10} **Traffic** (pkts/s)	R_{20} **Traffic** (pkts/s)	Other R_i **Average Traffic** (pkts/s)
10.0	65.66	38.27	89.16	120.41	8.95
20.0	85.15	49.38	85.05	115.84	8.54
30.0	92.26	51.03	81.85	113.30	8.75
40.0	94.55	52.47	75.79	102.12	9.67
50.0	98.77	53.58	74.35	100.29	9.90

6 Related Work and Implementation Issues

One major security problem of the IP protocol is that the source IP address can be spoofed by users [3]. There is no provided means to discover the true origin of a packet. Various researchers have proposed different approaches to solve the DDoS attack problem, including ingress filtering [7], link testing [4], logging [16], and throttling [21]. In link testing, we interactively test a router's upstream link starting from the nearest routers. The process is repeated until the source of an attack is found. Stone [19] proposes an automatic way to trace attackers within their own networks. The communication overhead in their approach for exchanging messages is very high and interoperations are difficult due to the existence of different administrative domains. Burch and Cheswick [4] propose another approach called controlled flooding. In controlled flooding, a victim site interactively floods its upstream links starting from the nearest routers. Then the victim site observes the changes in the rate of attack packets and identifies the attack packet origins. Selective flooding is repeated for farther routers until the full attack path is found. Controlled flooding consumes significant network resources including network bandwidth. If it is not handled or implemented carefully, the approach can itself become a form of network attack.

Logging is an 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. A full attack path can be reconstructed by applying certain data extraction methods. An advantage of the approach is that it does not increase the volume of traffic. However, it increases the storage requirement at the participating routers. Snoeren et al. [16]

TABLE 8
Percentage of Local Traffic to the Total Received Traffic by $\mathcal{V}$ after 50.0 Seconds

Router	**% of local to total traffic**	**Router**	**% of local to total traffic**	**Router**	**% of local to total traffic**	**Router**	**% of local to total traffic**
R_1	2%	R_6	2%	R_{11}	1%	R_{16}	2%
R_2	3%	R_7	1%	R_{12}	2%	R_{17}	2%
R_3	2%	R_8	2%	R_{13}	3%	R_{18}	2%
R_4	**14%**	R_9	5%	R_{14}	2%	R_{19}	5%
R_5	4%	R_{10}	**19%**	R_{15}	2%	R_{20}	**25%**

propose an efficient hash-based technique for storing *packet digests*. They claim that the storage requirement is less than one percentage of the link capacity per unit time. However, because tens of thousands of packets can traverse a router every second, the logging data can still grow quickly to an enormous size, which is especially problematic for very high speed links. Steven and Bellovin [18] propose ICMP traceback (later extensions include Wu et al. [20], [10]) which can traceback the attackers without incurring much overhead at the routers. In the approach, routers probabilistically generate an authenticated copy of a packet, including information about the adjacent routers along the path to the destination. The information can be used to reconstruct the path to the attackers. However, ICMP traffic can be easily identified and may be blocked or rate limited. ICMP traceback also needs key distribution and management to avoid the faking of ICMP packets.

Savage et al. [14] propose an elegant probabilistic marking method for traceback without generating separate ICMP packets to the victim site $\mathcal{V}$. Their approach does not require coordination among the network administrators and does not increase the network traffic or the storage requirement of a router. Dean et al. [6] formulate the traceback problem as a polynomial reconstruction problem. They use algebraic coding theory to encode traceback information in a packet, similar to Savage et al. [14]. However, their approach also suffers the packet spoofing problem and may even be more vulnerable without the distance field in the marking. Song and Perrig [17] report that if the victim site $\mathcal{V}$ knows the map of its upstream routers, it does not need the full IP address in the packet marking. They improve on the marking approach in [14] by hashing so as to achieve a lower false positive rate and a lower computation overhead. They propose efficient authentication of packet markings to filter packets with spoofed markings from the attackers. Note that the approaches in [14] and [17] provide the topology of the attack graph only. Our approach can be viewed as a complementary approach to theirs in also locating the potential attackers in an attack graph.

To realize the proposed traceback method, some further implementation issues need to be considered. For example, one needs routers to support the probabilistic traceback functions. If a router, say R_j, does not support packet marking and does not participate in the traceback process, then all its local traffic will be treated as the local traffic of its first downstream router which participates in the marking process. Therefore, as long as one router provides this service, the proposed traceback methodology will be able to perform attack tracing up to that point and enable packet filtering. Of course, under this situation, legitimate packets may be filtered out as well. If more routers participate in the marking and traceback process, the packet filtering can be more accurate and effective. This is an important point because one can incrementally deploy our proposed scheme in the Internet. Another issue is the computational complexity of estimating local traffic at all the routers. The computational complexity occurs mainly in (8), and it can be shown [9] that the worst case complexity is $O(n^2)$. Note that the proposed method does *not* depend on a particular attack signature, but rather on the intensities of the attack traffic. Therefore, when a victim site is under attack, either the victim's resources are overwhelmed or the bandwidth of the input link to the victim site is overwhelmed and the victim site requests all participating routers to perform probabilistic edge marking. Based on the collected marked packets, the proposed solution *deduces* the local traffic rates of all the routers. The purpose is to find out those routers whose local traffic consumes a sufficiently large percentage of the victim's resources.

Note that the traceback methodology we propose in this paper assumes that the attack traffic volume is large in the attack path. However, this is not always the case when many machines are compromised and each machine simply sends a small amount of attack traffic to the victim site. In this case, the distributed throttling approach proposed in [21], [22] is effective since it guarantees the "*level-k max-min fairness*" to upstream traffics.

7 Conclusion

In this paper, we have considered 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 is based on a probabilistic packet marking algorithm: A victim site $\mathcal{V}$, upon discovering that it is being attacked, requests a set of routers to mark all the packets addressed to $\mathcal{V}$ with a given probability p. Based on the collected marked packets, the victim site $\mathcal{V}$ can 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 signal the relevant routers (i.e., those routers which consume a high fraction of the victim's resources) to eliminate their local traffic. An important technical contribution is our 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. We have carried out experiments to illustrate the effectiveness and robustness of our algorithm under both linear and general network topologies. We have shown the robustness of our estimation when the packet arrival process takes on different forms, including Poisson, constant rate, bursty rate, and random on/off. Our results show that we can locate the attackers in a short time. The proposed methodology provides a useful step towards an automated wide-area network traceback facility.

Acknowledgments

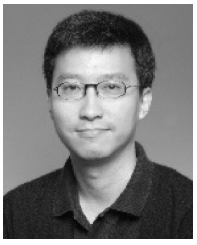
This research was supported in part by a Hong Kong Government Earmarked Grant, by the US National Science Foundation under grant numbers EIA-9806741 and CCR-9875742, by CERIAS, and by the Indiana 21st Century Research and Technology Fund.

References

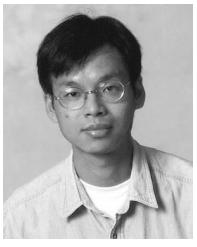
[1] Internet scanning database, http://cm.bell-labs.com/who/ches/map/dbs/index.html, 1999.

[2] "Computer Emergency Response Team, Cert Advisory ca-2000-01: Denial-of-Service Developments," http://www.cert.org/advisories/ca-2000-01.html, 2000.
[3] S. Bellowin, "Security Problems in the TCP/IP Protocol Suite," *Computer Comm. Rev.*, pp. 32-48, 1989.
[4] H. Burch and B. Cheswick, "Tracing Anonymous Packets to their Approximate Source," *Usenix LISA*, Dec. 2000.
[5] R.L. Burden and J.D. Faires, *Numerical Analysis.* Boston: PWS-Kent Publishing Company, 1988.
[6] D. Dean, M. Franklin, and A. Stubblefield, "An Algebraic Approach to IP Traceback," *Proc. Network and Distributed System Security Symp. (NDSS '01)*, Feb. 2001.
[7] P. Ferguson and D. Senie, "RFC 2267: Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing," *The Internet Society*, Jan. 1998.
[8] J. Howard, "An Analysis of Security Incidents on the Internet," PhD Thesis, Carnegie Mellon Univ., Aug. 1998.
[9] 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," Technical Report CS-TR-2002-06, Computer Science Eng. Dept., Chinese Univ. Hong Kong, 2002.
[10] A. Mankin, D. Massey, C.-L. Wu, S.F. Wu, and L. Zhang, "On Design and Evaluation of Intention-Driven ICMP Traceback," *Proc. IEEE Int'l Conf. Computer Comm. and Networks*, 2001.
[11] R. Nelson, *Probability, Stochastic Processes, and Queueing Theory: The Mathematics of Computer Performance Modeling.* Springer-Verlag, 1995.
[12] T. Peng, C. Leckie, and R. Kotagiri, "Adjusted Probabilistic Packet Marking for IP Traceback," *Proc. Conf. Networking*, May 2002.
[13] S.M. Ross, *Stochastic Processes.* John Wiley, 1996.
[14] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical Network Support for IP Traceback," *Proc. 2000 ACM SIGCOMM Conf.*, pp. 295-306, Aug. 2000.
[15] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Network Support for IP Traceback," *ACM/IEEE Trans. Networking*, vol. 9, no. 3, June 2001.
[16] A.C. Snoeren, C. Partridge, L.A. Sanchez, C.E. Jones, F. Tchakountio, S.T. Kent, and W.T. Strayer, "Hash-Based IP Traceback," *Proc. ACM SIGCOMM 2001 Conf. Applications, Technologies, Architectures, and Protocols for Computer Comm.*, Aug. 2001.
[17] D.X. Song and A. Perrig, "Advanced and Authenticated Marking Schemes for IP Traceback," *Proc. IEEE INFOCOM Conf.*, Apr. 2001.
[18] E. Steven and M. Bellovin, "ICMP Traceback Messages," draft-bellovin-itrace-00.txt, Mar. 2000.
[19] R. Stone, "Centertrack: An IP Overlay Network for Tracking DOS Floods," *Proc. Ninth USENIX Security Symp.*, Aug. 2000.
[20] S.F. Wu, L. Zhang, D. Massey, and A. Mankin, "Intention-Driven ICMP Trace-Back, Internet Draft," draft-wu-itrace-intention-00.txt, Feb. 2001.
[21] 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," *Proc. IEEE Int'l Workshop Quality of Service (IWQoS)*, May 2002.
[22] D.K.Y. Yau, J.C.S. Lui, F. Liang, and Y. Yeung, "Defending against Distributed Denial-of-Service Attacks with Max-Min Fair Server-Centric Router Throttles," *Proc. 10th IEEE Int'l Workshop Quality of Service*, 2002.

Terence K.T. Law received the BSc and MPhil degrees in computer science from the Chinese University of Hong Kong (CUHK). He is presently a researcher at CUHK. His research interests span in network communication and security. His personal interests include films, sports, and general reading.



John C.S. Lui received the PhD degree in computer science from the University of California at Los Angeles. After graduation, he joined the IBM Almaden Research Laboratory/San Jose Laboratory and participated in various research and development projects on file systems and parallel I/O architectures. He later joined the Department of Computer Science and Engineering at the Chinese University of Hong Kong. His research interests span both in system and in theory/mathematics. His current research interests are in theoretic/applied topics in data networks, distributed multimedia systems, network security, OS design issues, and mathematical optimization and performance evaluation theory. He received the CUHK Vice-Chancellor's Exemplary Teaching Award in 2001. He is an associate editor of the *Performance Evaluation Journal*, a member of ACM, a senior member of IEEE, a member of the IEEE Computer Society, and an elected member in the IFIP WG 7.3, the TPC cochair for the ACM Sigmetrics 2005. His personal interests include films and general reading.



David K.Y. Yau received the BSc (first class honors) degree from the Chinese University of Hong Kong and the MS and PhD degrees from the University of Texas at Austin, all in computer sciences. From 1989 to 1990, he was with the Systems and Technology group of Citibank. He was the recipient of an IBM graduate fellowship and is currently an Associate Professor of Computer Sciences at Purdue University, West Lafayette, Indiana. He received an NSF CAREER award in 1999 for research on network and operating system architectures and algorithms for quality of service provisioning. His other research interests are in network security, value-added services routers, and mobile wireless networking. He is a member of the IEEE Computer Society.