A Divide-and-Conquer Strategy for Thwarting Distributed Denial-of-Service Attacks

Ruiliang Chen, *Student Member*, *IEEE*, Jung-Min Park, *Member*, *IEEE*, and Randolph Marchany, *Member*, *IEEE*

Abstract—Attack mitigation schemes actively throttle attack traffic generated in Distributed Denial-of-Service (DDoS) attacks. This paper presents *Attack Diagnosis* (AD), a novel attack mitigation scheme that adopts a divide-and-conquer strategy. AD combines the concepts of Pushback and packet marking, and its architecture is in line with the ideal DDoS attack countermeasure paradigm—attack detection is performed near the victim host and packet filtering is executed close to the attack sources. AD is a reactive defense mechanism that is activated by a victim host after an attack is detected. By instructing its upstream routers to mark packets deterministically, the victim can trace back one attack source and command an AD-enabled router close to the source to filter the attack packets. This process isolates one attacker and throttles it, which is repeated until the attack is mitigated. We also propose an extension to AD called *Parallel Attack Diagnosis* (PAD) that is capable of throttling traffic coming from a large number of attackers simultaneously. AD and PAD are analyzed and evaluated using the Skitter Internet map, Lumeta's Internet map, and the 6-degree complete tree topology model. Both schemes are shown to be robust against IP spoofing and to incur low false positive ratios.

Index Terms—Network-level security and protection.

1 Introduction

Large-scale, high-profile Distributed Denial-of-Service (DDoS) attacks have become common recurring events that increasingly threaten the proper functioning of the Internet. Despite a significant breadth of research into countermeasures, DDoS attacks remain a major threat today.

Defending against DDoS attacks is challenging for two reasons. First, the number of attackers[1] involved in a DDoS attack is very large. Even if the volume of traffic sent by a single attacker might be small, the volume of aggregated traffic arriving at the victim host is overwhelming. Second, attackers usually spoof their IP addresses, which makes it very difficult to trace the attack traffic back to its sources. Though *ingress/egress filtering* [16] has been deployed in many subnets to prevent IP spoofing, their effectiveness is sometimes limited. Moreover, ingress/egress filtering does not prevent *subnet spoofing* [24]. In subnet spoofing, an attacker spoofs a random address from the address space assigned to its subnet.

DDoS attack countermeasures can be categorized into four classes: prevention, detection, mitigation, and traceback (response) [8]. Mitigation techniques can be divided into two categories. The schemes in the first category regard DDoS defense as a resource allocation problem. They employ techniques such as *client puzzles* [9], [14], [21], [35], *max-min servercentric router throttles* [38], or *differentiated service* [36] to allocate network or server resources to clients in a fair fashion, thus preventing attackers from consuming an excessive amount of network resources. These schemes can effectively suppress attackers that generate traffic at a high rate. However, since they do not distinguish legitimate clients from malicious ones, high-rate legitimate traffic may be throttled, causing "collateral damage." More importantly, they are not effective against low-rate DDoS attacks [4], [19]. The mitigation techniques in the second category actively thwart DDoS attacks by filtering or rate-limiting attack packets. A scheme belonging to this category usually consists of two modules: an *attack detection* module and a *packet filtering* module. The attack detection module is used to extract the characteristics of attack packets, i.e., "attack signatures," such as source IP addresses or marked IP header values [33], [39]. After the characteristics have been summarized, this information is used by the packet filtering module to filter malicious packets. The mitigation schemes presented in [22], [26], [33], [34], [37], [39] place the two modules close to each other, i.e., they are placed close to the victim [33], [34], [37], [39], [40], in the core network [26], or close to the attack sources [22]. However, placing the two modules at the same location limits their effectiveness. Attacks can be most effectively detected at the victim end, where all the attack packets can be observed readily. In contrast, it is most effective to filter attack packets as close to the attack sources as possible, thus preventing the attack traffic from reaching deeper into a network. Therefore, in the ideal countermeasure paradigm, the attack detection module is placed at (or near) the victim and the packet

1. We use the term "attacker" to refer to a zombie machine rather than its controller.

- *R. Chen and J.M. Park are with the Bradley Department of Electrical and Computer Engineering, Virginia Polytechnic and State University, 302 Whittemore Hall, Blacksburg, VA 24061. E-mail: {rlchen, jung-min}@vt.edu.*
- *R. Marchany is with the Information Technology Security Office, Virginia Polytechnic Institute and State University, 1300 Torgersen Hall, Blacksburg, VA 24061. E-mail: Marchany@vt.edu.*

Manuscript received 7 Nov. 2005; revised 25 Apr. 2006; accepted 29 June 2006; published online 9 Jan. 2007.
Recommended for acceptance by G. Lee.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0462-1105.
Digital Object Identifier no. 10.1109/TPDS.2007.1014.

filtering module is placed as close to the attack sources as possible. For more details on this paradigm, see [2], [23].

Schemes proposed in [6], [20], [27], [44] support this paradigm. In these schemes, when attacks are detected downstream close to the victim, the upstream routers close to the attack sources filter attack packets using summarized attack signatures sent by the detection module. These schemes, however, have either or both of the following two drawbacks: The first drawback is the need to securely forward attack signatures to the upstream routers. The schemes proposed in [27] and [44] require a global key distribution infrastructure for authenticating and verifying the attack signatures. Such an infrastructure is costly to deploy and maintain. The second drawback is the dependence on attack signatures to separate attack traffic from legitimate traffic. Using such a signature is very difficult for three reasons. First, in many cases, an attack detection module can only detect the existence of attacks but may not formulate any attack signatures from the observed traffic [8]. Second, even if an attack signature is obtained, it may not be usable to a router to filter packets when the signature lies above the network layer. Because a router is a network-layer device, it would severely degrade its performance to examine the contents of every packet to match high-layer attack signatures (e.g., TCP SYN packets, HTTP requests with junk cookie payload, etc.). Last, even signatures in the network layer may have limited value because the attackers can readily manipulate corresponding information. For example, an attacker can spoof source IP addresses and change the *protocol* field in the IP header, rendering these fields useless as valid attack signatures. The only absolutely reliable information in a packet is the destination IP address. However, packet filtering based solely on the destination IP address will throttle legitimate traffic, causing significant collateral damage.

In this paper, we propose *Attack Diagnosis (AD)*, a novel attack mitigation scheme that combines the concepts of Pushback [20] and packet marking to thwart DDoS attacks. AD takes a "divide-and-conquer" strategy in the sense that it consists of a repetitive process of isolating an attacker and then filtering its traffic. The execution of AD can be summarized in four steps:

1. An Intrusion Detection System (IDS) installed at the victim (or at its firewall) detects an attack.
2. The victim instructs the upstream routers to start marking packets with traceback information.
3. Based on the marking information extracted from collected packets, the victim separates an attacker from other clients and traces back to the attack source.
4. The victim instructs the appropriate upstream routers to filter attack packets.

These four steps compose one *round* of AD, which isolates one attacker from others and throttles its traffic. AD repeats this process for multiple rounds until the attack is mitigated. This technique can be employed to thwart attacks involving a moderate number of attackers. However, AD is not appropriate for large-scale attacks involving a large number of attackers because the process would be very slow. To address this problem, we propose an extension to AD called *Parallel Attack Diagnosis (PAD)* that can throttle traffic coming from multiple attackers simultaneously in a single round. PAD can significantly accelerate the attack mitigation process at the cost of increased false positives.

Our approach has the following noteworthy features. To the best of our knowledge, no existing DDoS countermeasure possesses all of the features listed below:

- AD and PAD support the ideal DDoS countermeasure paradigm, in which the attack detection module is placed close to the victim while the packet filtering module is located close to the attackers.
- Our approach is reactive in nature. No communication overhead is required when a network is not under attack.
- Because AD and PAD employ *deterministic* packet marking, they are robust against forgery of marking fields, which plagues probabilistic packet marking schemes [7], [11], [28], [31], [42].
- Because AD and PAD throttle attackers in a "divide-and-conquer" fashion, very low false positive ratios are incurred. Moreover, PAD provides a "tunable" parameter that enables the network to adjust the diagnosis process delay and the false positive ratio.
- AD and PAD do not rely on attack signatures for packet filtering. When AD and PAD are deployed in a spatially contiguous manner, they do not require a global key distribution infrastructure. These properties are of practical importance.

The rest of the paper is organized as follows: In the next section, we elaborate AD and PAD. In Section 3, we discuss practical considerations. The simulation results are presented in Section 4. We survey related work in Section 5 and conclude the paper in Section 6.

2 Attack Diagnosis and Parallel Attack Diagnosis

2.1 Assumptions

We assume the following network environment: Every host, either a client or a server, is connected to its local *edge router*.[2] Edge routers are, in turn, interconnected by *core routers*. We refer to the server host being attacked as the *victim*. A recent study [13] has shown that 95 percent of the routes observed in the Internet have fewer than five observable daily changes. So, we make the reasonable assumption that every route from a client to the victim is fixed during the timeframe of interest. We also assume that Internet routers are not compromised.

We use the term *false negative* to denote an attacker whose malicious packets have not been filtered, and we use the term *false positive* to denote a legitimate client whose packets have been incorrectly throttled.

Like other packet-marking-based mitigation schemes [33], [41], we assume the existence of an IDS module installed at the victim (or at its firewall), which is able to identify the existence of attacks after observing malicious traffic.

2. The mechanisms of AD/PAD are transparent to link-layer devices such as switches. Therefore, any switches that may be connected between a host and its edge router are ignored in the discussion of network environment.

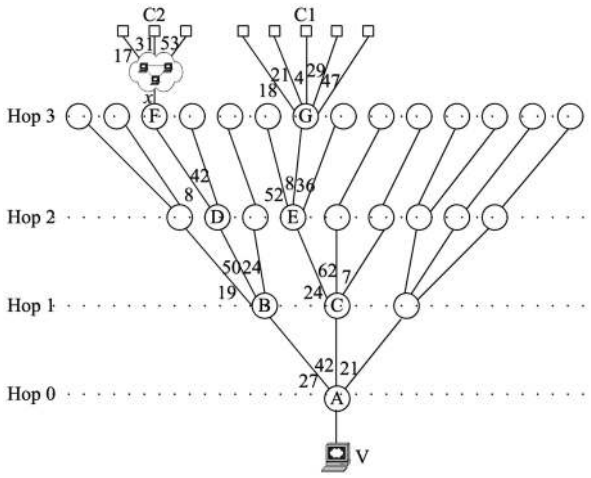

Fig. 1. An upstream tree of victim V.

2.2 Overview of AD and PAD

We will illustrate the principles of AD and PAD using Fig. 1. This figure shows an upstream tree of victim V. In the figure, some of the router interfaces are labeled with a locally unique number that identifies that interface port. We call this number the *port identifier (PID)*. PID is locally unique in the sense that interfaces of two different routers can have the same PID, while the interfaces of a single router are assigned nonrepeating PIDs.

In most cases, a PID of a router can be used to uniquely identify a router or a host that is connected to it. However, when an interface port is connected to multiple hosts via a broadcast link-layer channel (such as in a LAN), a PID cannot be used to uniquely identify a host. An example of such a case is shown in Fig. 1. In the figure, interface x of router F is connected to multiple clients through a LAN. In this case, router F maintains a *virtual PID table* that maps each "virtual" interface to a MAC address. More precisely, the table maps a "virtual PID" to every MAC address that the router observes coming through interface x. For example, in Fig. 1, the MAC address of $C2$'s Ethernet adapter is mapped to the virtual PID 31.

Since a PID is locally unique within a router, a string of PIDs can be used to uniquely identify the path from a server to a client. For example, in Fig. 1, the string 4-8-24-42 (i.e., the PIDs corresponding to each hop from hop 3 to hop 0) represents the path from $C1$ to V. If $C1$ is an attacker, this string will represent an attack path. Hence, constructing a PID string corresponds to reconstructing an attack path in this instance. AD uses an iterative process to construct such a PID string, starting with PID 42. Once the reconstruction of the attack path is complete, the router closest to the attacker (i.e., router G) filters all packets destined for V at interface 4.

Although AD is capable of throttling malicious traffic coming from a modest number of attackers, it does not reconstruct attack paths fast enough to effectively mitigate attack traffic coming from a large number of attackers. PAD solves this problem by dealing with multiple attackers simultaneously.

2.3 Attack Diagnosis

To support AD, a router needs to mark packets with its PIDs and other traceback-related information. For this purpose, we overload the 16-bit *Identification* field and one reserved bit in the IP header. All existing probabilistic packet marking schemes overload the *Identification* field. The justification for overloading this field is based on the observation that IP fragments constitute a very small proportion of the actual Internet traffic (less than 0.25 percent) [7], [28]. We use an a-bit *hop-count* field, a b-bit *PID* field, and a c-bit *XOR* field, where $a + b + c = 17$ and $b \geq c$. A discussion on how to choose the values of a, b, and c will be discussed in Section 3.1. In this section, for the convenience of discussion, we assume $a = 5$, $b = 6$, and $c = 6$. The *hop-count* field in a packet records the number of hops from a router that first marks a given packet to the edge router that is immediately upstream of the victim. The *PID* field of a given packet records the PID of the router's input interface port that processed the packet. The *XOR* field of a packet records the value obtained by taking the XOR (exclusive OR) of the least significant c bits of PID values. AD does not use the *XOR* field, but PAD uses it for distinguishing different attack paths.

A router interface can be set to either of two marking modes. A router interface is said to be in the Active Deterministic Marking Mode (ADMM) for V if it processes every packet destined for V as follows: 1) sets the *hop-count* field to zero, 2) copies its PID to the *PID* field, and 3) copies the least significant c bits of its PID to the *XOR* field. A router interface is in the Passive Deterministic Marking Mode (PDMM) for V if it processes every packet destined for V as follows: 1) increases the *hop-count* field by one and 2) computes the bit-by-bit XOR value of the least significant c bits of its *PID* and the *XOR* field value and writes the result back to the *XOR* field.

When the IDS installed at the victim detects an attack, the AD process is triggered. The victim begins the process by sending a "Diagnose-All-Interfaces" (DAI) request to its immediate edge router. This request packet should have the *TTL* field set to 255 so that the receiving router will be able to verify that the packet came from a host one hop away. Refer to Fig. 1 for an example. The path $C1 - G - E - C - A$ is assumed to be the attack path of a single attacker, $C1$. The execution of AD is summarized in the following steps:

1. First, V sends a DAI request to router A. Reacting to the request, A sets the marking mode of all of its input interfaces to ADMM for V. Also, A sends a status packet to V to notify V that it has begun marking packets. Now, every packet arriving at V has its *hop-count* field marked as zero and its *PID* field marked as the PID value of A's interface that processed the packet. If the IDS is able to detect an ongoing attack based on the observation of received packets, then V should be able to identify the input interfaces of A that processed the malicious packets. One method for carrying this out is to group the received packets based on their *PID* markings and then identify the interfaces by selecting a group that contains suspicious packets.

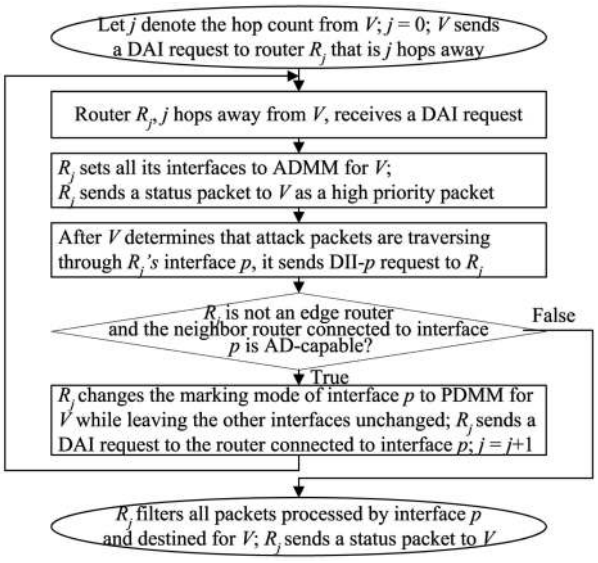

Fig. 2. A flowchart representation of AD execution.

2. After V has observed that the attack traffic is coming from the interface 42 of A, it sends a "Diagnose-Individual-Interface" 42 (DII-42) request to A. A router accepts a DII request for a victim only when a DAI request for the same victim was received previously. The DII request is accepted because A just received a DAI request for V. Otherwise, this request should be ignored. Router A executes two steps to respond to the DII request. First, it sets the interface of PID 42 to PDMM while leaving the other interfaces unchanged. Second, A sends a DAI request to the neighbor router connected via interface 42, namely, router C in our running example. Again, this request packet's *TTL* field is set to 255.
3. Router C executes the same steps that A did when it received the DAI request from V. Again, a status packet is sent to V. This packet notifies V of C's IP address and the fact that C is marking packets. Now, packets received by V with their *hop-count* fields equal to one are all coming from C. Using this information, V is able to identify the input interface of C that is processing the attack traffic, which is interface 24 in our running example.
4. After identifying the interface in the previous step, V sends to C a DII-24 request. C executes the same procedures executed by A in Step 2.
5. The procedures described in the previous steps are iterated until router G receives a DII-4 request. After this request is received, G begins to filter all the packets destined for V that are processed by interface with PID 4. If interface 4 is connected to multiple hosts via a broadcast link-layer channel, then G refers to the virtual PID table and filters frames based on the source MAC address. Router G also sends a status packet to V to notify V that it has started the filtering process.

The first four steps constitute the traceback phase, and the last step is the filtering phase. The five steps constitute a round of diagnosis. With these steps, AD can effectively separate $C1$ from other clients, trace back to its source, and filter its attack packets. The above steps are summarized in Fig. 2 as a flowchart. When there are multiple attackers, these steps are repeated with each step separating and throttling one attacker until the attack is mitigated.

The legitimacy and integrity of the DII and DAI control messages need to be assured in order for AD to work properly. This particular security problem will be discussed in Section 3.

2.4 Parallel Attack Diagnosis

AD traces back and throttles the traffic of one attacker at a time. It is obvious that this technique is too slow to defend against a large-scale DDoS attack in which (tens of) thousands of attackers generate attack traffic synchronously. In such a case, the victim may be inundated with millions of attack packets before AD brings about any noticeable effect. Therefore, we introduce a "parallelized" version of AD—Parallel Attack Diagnosis (PAD)—that can handle multiple attack paths simultaneously. The primary difference between PAD and AD is that, in PAD, when a node sends a DII request to a router, it can specify more than one PID. The router that receives the DII request changes the marking mode of the appropriate interfaces (which are specified by the PIDs) to PDMM and sends DAI requests to the upstream neighbor routers that are connected to the interfaces. If the router is an edge router connected to hosts, then it simply begins to filter attack packets at the interfaces.

To function properly, PAD needs to distinguish distinct attack paths that are being traced back simultaneously. To illustrate this point, we refer back to Fig. 1. Assume that there are two attackers, $C1$ and $C2$, attacking V at the same time. After sending a DAI request to A, V observes that there are two interfaces, 42 and 27, receiving malicious packets. If V decides to "diagnose" the two interfaces in parallel, it sends a "DII-42, 27" request to A. Then, A will change the marking mode of the two interfaces to PDMM, and send DAI requests to both B and C. In response, B and C will set the marking mode of all their interfaces to ADMM. This scenario raises an important question: When parsing through packets that have the *hop-count* field set to one, how can V determine that interface 50 belongs to B and interface 24 belongs to C? This problem can be solved by using the *XOR* field. Recall that the *XOR* field of a packet contains the XOR result of the least significant c bits of PIDs (we assume $b = c = 6$ in the discussions) whose corresponding interfaces marked that packet along a path. Hence,

$$\begin{aligned} &XOR(j) \oplus PID(j) = \\ &PID(0) \oplus \ldots \oplus PID(j) \oplus PID(j) = \\ &PID(0) \oplus PID(1) \oplus \ldots \oplus PID(j-1) = \\ &XOR(j-1), \end{aligned} \tag{1}$$

where $XOR(j)$ and $PID(j)$ denote the bit strings of the respective marking fields of a packet at the instant that the packet is marked in ADMM j hops away from the victim, and $\oplus$ denotes the XOR operation. Using (1), the victim can group the PIDs that constitute an attack path. In our running example, PID 50 at hop 1 is grouped with PID 27 (and not PID 42) at hop 0 because $41 \oplus 50 = 27$, where 41 is the value of the *XOR* field. In the same manner, PID 24 at hop 1 can be grouped with PID 42 at hop 0. This process is repeated for every hop until a DII request is received by the edge routers, G and F. In response, the routers filter packets based on the last-hop PID specified in the DII message. Table 1 shows how the PIDs can be grouped to form an attack path. The shaded rows correspond to the attack path of $C1$, and the nonshaded rows correspond to the attack path of $C2$.

TABLE 1
Using the *XOR* Field to Differentiate Multiple Attack Paths

Hop-count	*	PID	XOR
0	A	42 [101010]#	42 [101010]
	A	27 [011011]	27 [011011]
1	C	24 [011000]	50 [110010] ($\oplus$24 = 42)
	B	50 [110010]	41 [101001] ($\oplus$50 = 27)
2	E	08 [001000]	58 [111010] ($\oplus$08 = 50)
	D	08 [001000]	33 [100001] ($\oplus$08 = 41)
3	G	04 [000100]	62 [111110] ($\oplus$04 = 58)
	F	31 [011111]	62 [111110] ($\oplus$31 = 33)

** The farthest router from V that marks packets in ADMM.*
 # The binary string inside [ ] represents the binary equivalent.

2.5 Analysis of False Positives

In this and the next sections, we analyze false positives and attack mitigation delay in AD and PAD. The attack mitigation delay is the interval between the start time of attack detection and the termination time of AD/PAD.[3] We shall show that there is a trade-off between the two metrics and that PAD is capable of tuning its parameters to balance the two metrics.

Under the following assumptions, one can show that AD incurs no false negatives or false positives and PAD incurs no false negatives: 1) the attackers keep sending attack packets to the victim during the AD or PAD process, 2) the IDS installed at the victim can accurately identify attacks, and 3) the MAC addresses are not spoofed. Unlike AD, PAD handles multiple attackers simultaneously in a single round of diagnosis. This can result in false positives. The reason is that, during the traceback phase, the *XOR* and *PID* fields of the packets coming from a legitimate client may coincide with those coming from an attacker. For example, a PID string corresponding to a legitimate client's path, $P_{12} - P_{11} - P_{10}$, can incur a false positive when the substring $P_{11} - P_{10}$ is a substring of an attacker's path and there is another attacker's path $P_{22} - P_{21} - P_{20}$ under diagnosis, which satisfies the following two conditions: 1) $P_{22} = P_{12}$ and 2) $(P_{10} \oplus P_{11})_c = (P_{20} \oplus P_{21})_c$. Here, $(x)_c$ denotes the least significant c bits of x. The first condition leads to a *collision* of the *PID* fields and the second condition results in a collision of the *XOR* fields. Generalizing the above argument, a false positive occurs when each link of a legitimate client's path (that is not on any attack path) collides with a link of an attack path, in terms of both the *XOR* field and the *PID* field. Allocating PIDs randomly can minimize the chance of collision. Hence, we assume that each router assigns PID values to its interfaces randomly from 0 to $2^b - 1$. If we assume that q attackers are diagnosed in a single round, then the probability that a link of a legitimate client's path (that is not on any attack path) collides with a link on an attack path is

$$P_C(q) = \frac{\sum_{k=1}^{\min(q,2^{b+c})} k \binom{2^{b+c}}{k} \left(\frac{1}{2^{b+c}}\right)^q M(k)}{2^{b+c}} (q \geq 2), \tag{2}$$

where

$$M(k) = \begin{cases} 1 & (k = 1) \\ k^q - \sum_{l=1}^{k-1} \binom{k}{l} M(l) & (2 \leq k \leq 2^{b+c}). \end{cases}$$

3. Note that, in practice, AD/PAD may terminate when the incoming rate of malicious traffic drops below a certain threshold.

If a legitimate client's path has m links that do not coincide with any links on any attack path, then its false positive probability is

$$P_{FP}(q) = [P_C(q)]^m. \tag{3}$$

The value given by (3) represents the false positive probability for a single round of diagnosis. Because q attackers are diagnosed per round, PAD executes a total of $\lceil \frac{N_a}{q} \rceil$ rounds, where $N_a (\geq q)$ is the total number of attackers to be throttled. Therefore, the probability of incurring a false positive in PAD is

$$\begin{aligned} &P_{FP-PAD}(N_a, q) = \\ &\quad 1 - [1 - P_{FP}(q)]^{\lfloor \frac{N_a}{q} \rfloor} \cdot [1 - P_{FP}(N_a \bmod q)]. \end{aligned} \tag{4}$$

For example, if $N_a = 1{,}000$, $q = 100$, and $m = 2$, we obtain $P_{FP} = 0.059\%$ and $P_{FP-PAD} = 0.59\%$. In reality, the value of m is not fixed and is very difficult to estimate. In Section 4, we will present figures that plot (4) as a function of q using simulation data.

2.6 Analysis of Attack Mitigation Delay

In this section, we discuss the attack mitigation delay incurred by AD and PAD. We first define the following variables:

$H(i)$. The hop count of the ith attack path, where $i = 1, \ldots, N_a$.

Q. The number of malicious packets the victim's IDS needs to sample before being able to detect the existence of attacks.

$R(i, j)$. The aggregate arrival rate of malicious packet rate at the j-hop router (away from the victim's edge router) on the ith attack path arriving at the victim, where $j = 0, \ldots, H(i) - 1$.

$T_{dec}(i, j)$. The time required to detect attacks aggregated at the j-hop router on the ith attack path. When the IDS's processing time is ignored, we have $T_{dec}(i, j) = Q/R(i, j)$.

$T_{pro}(i, j)$. The time needed by the j-hop router on the ith attack path to change its marking mode or filtering status.

$D(i, j)$. The path delay between the victim and the j-hop router on the ith attack path (including transmission delay, propagation delay, queuing delay, and processing delay for routing).

$D_1(i, j)$. The one-hop delay for the j-hop router on the ith attack path to send a DAI request to its neighboring upstream router.

Referring to Fig. 2 and assuming the ith attack path is being diagnosed, we can infer that the first step (the top oval) takes time $[T_{dec}(i, 0) + D(i, 0)]$ to finish, and each loop needs time

$$\begin{aligned} T(i) = &\ T_{pro}(i, j) + D(i, j) + T_{dec}(i, j + 1) + D(i, j) \\ &+ T_{pro}(i, j) + D_1(i, j) \\ = &\ 2T_{pro}(i, j) + 2D(i, j) + D_1(i, j) + \frac{Q}{R(i, j + 1)}. \end{aligned} \tag{5}$$

The last step (the bottom oval) takes time

$$T_{pro}(i, H(i) - 1) + D(i, H(i) - 1).$$

Therefore, the attack mitigation delay of AD, which is the sum of the time required for diagnosing all attack paths, is

$$\begin{aligned} T_{AD} = & \sum_{i=1}^{N_a} \Bigg\{ T_{dec}(i,0) + D(i,0) \\ & + \sum_{j=0}^{H(i)-1} \left[2T_{pro}(i,j) + 2D(i,j) + \frac{Q}{R(i,j+1)} + D_1(i,j) \right] \\ & + T_{pro}(i,H(i)-1) + D(i,H(i)-1) \Bigg\} \\ = \; & N_a[T_{dec}(1,0) + D(1,0)] \\ & + \sum_{i=1}^{N_a} \left[T_{pro}(i,H(i)-1) + D(i,H(i)-1) \right] \\ & + \sum_{i=1}^{N_a} \sum_{j=0}^{H(i)-1} \Bigg[2T_{pro}(i,j) + 2D(i,j) \\ & + \frac{Q}{R(i,j+1)} + D_1(i,j) \Bigg]. \end{aligned} \tag{6}$$

The attack mitigation delay of PAD can be computed similarly. However, because PAD synchronizes the diagnosis of q attack paths, the time taken in each hop's diagnosis will be the greatest time that it takes to diagnose among the q attack paths. This can be expressed as follows:

$$\begin{aligned} T_{PAD} = & \left\lceil \frac{N_a}{q} \right\rceil [T_{dec}(1,0) + D(1,0)] \\ & + \sum_{i=1}^{\lceil \frac{N_a}{q} \rceil} \left\{ \max_{i \in S(r)} \left[T_{pro}(i,H_S) \right] + \max_{i \in S(r)} [D(i,H_S)] \right\} \\ & + \sum_{i=1}^{\lceil \frac{N_a}{q} \rceil} \sum_{j=0}^{H_S} \Bigg\{ 2 \max_{i \in S(r)} \left[T_{pro}(i,j) \right] + 2 \max_{i \in S(r)} [D(i,j)] \\ & + \max_{i \in S(r)} \left[\frac{Q}{R(i,j+1)} \right] + \max_{i \in S(r)} [D_1(i,j)] \Bigg\}, \end{aligned} \tag{7}$$

where

$$S(r) = \{ i \mid (r-1)q + 1 \le i \le \min(N_a, rq), i \in \mathcal{N} \},^4$$

$$H_S = \max_{i \in S(r)} (H(i) - 1),$$

and when $j > H(i) - 1$, $T_{pro}(i,j)$, $D(i,j)$, $Q/R(i,j)$, and $D_1(i,j)$ are all equal to zero.

It can be seen from (4) and (7) that P_{FP-PAD} increases while T_{PAD} decreases as q is increased. Hence, a trade-off between the two metrics can be made by adjusting the value of q. In Section 4, we shall present figures that plot P_{FP-PAD} and T_{PAD} for various parameter settings.

3 Practical Considerations

3.1 Selection of Marking Field Length

We allocated a, b, and c bits for the *hop-count*, *PID*, and *XOR* fields respectively. In this section, we discuss how to set their values.

4. Here, $\mathcal{N}$ denotes the set of natural numbers.

In existing IP traceback schemes that employ probabilistic packet marking for [7], [11], [28], [31], [42], five bits are typically used to record the hop count. This is because the vast majority of routes in the Internet have fewer than 32 hops [13]. Therefore, we set $a = 5$, which implies $b + c = 12$ since the three fields combined occupy 17 bits. In AD, we allocate all the remaining 12 bits to the *PID* field, and the *XOR* field is not used. In PAD, because the c-bit *XOR* field takes at most b bits from the *PID* field for XOR operation, we have $b \ge c$. Now we argue that b should be minimized given that $b + c$ is fixed. In Section 2.5, it has been shown that the false positive probability is dependent only on the value of $b + c$ and not on the individual values of b and c. However, the collision of the b-bit *PID* field and that of the c-bit *XOR* field, which are the two necessary conditions to induce a false positive, have different implications. The role of the *XOR* field is to distinguish individual routes so that the DII commands can be issued to the appropriate routers. On the other hand, the *PID* field identifies an individual interface port on a router, which helps the router discard DII requests with wrong PIDs. DII requests with incorrect PIDs are received when the collision of the *XOR* field takes place. Therefore, in order to minimize the chance of collision of the *XOR* field, the length of the *XOR* field needs to be maximized. Hence, we set the field lengths as $a = 5$, $b = 6$, and $c = 6$.

CAIDA's (Cooperative Association for Internet Data Analysis) Skitter study [29] showed that 98.5 percent of Internet routers have fewer than 64 working interfaces. Another Internet topology study [1] shows that this percentage is even higher. Hence, allocating six bits for the *PID* field should be sufficient for the vast majority of Internet routers. However, six bits are not enough for routers with more than 64 working interfaces and edge routers connected to more than 64 hosts in an edge network. This problem can be solved by adding another six bits to a PID and employing the "enhanced" packet marking procedure. In the enhanced procedure, a router acts as if it was two routers connected in serial. The router splits the PID into two 6-bit PID fragments and associates them with different hops. For example, if B in Fig. 1 uses 12-bit PIDs, it associates the six most significant bits with hop 1 and the six least significant bits with hop 2. In each hop diagnosis, B still only marks six bits, but the diagnosis of the two hops reveals the complete 12-bit PID. As a result, D is considered to be located at hop 3 and F at hop 4.

The increased hop count caused by the enhanced packet marking scheme leads to a new problem. The five bits allocated to the *hop-count* field may not be enough to record the hop count of long routes. To avoid such a situation, we can allocate six bits to the *hop-count* field, i.e., $a = 6$, and set the other two fields as $b = 6$ and $c = 5$. In Section 4, we show the impact that the values of a, b, and c have on the performance of PAD, particularly in terms of false positive ratios.

3.2 Security Considerations

The DAI requests used in AD and PAD are secured in two ways. First, the core routers only accept DAI requests from neighboring routers, i.e., the packets sending DAI requests must have a *TTL* field set to 255. Second, the edge routers are responsible for authenticating DAI requests originating from end hosts. The key required to authenticate the DAI requests can be established during an offline registration

process. Therefore, forgery of DAI requests is thwarted. As for the DII requests issued by a victim, as Fig. 2 indicates, any DII request received by a router is always preceded by a DAI request. The DAI request is in turn triggered by another DII request from the victim that was sent to the router's downstream neighbor, which corresponds to the DII request of the previous loop in the flowchart of Fig. 2. Therefore, all DII requests form a chain in a round of diagnosis. The first DII request can be authenticated in the same way as the original DAI request, since they are both sent by the victim to its edge router. Now, if we can utilize a preceding DII request to authenticate its subsequent DII request in the chain, then all the DII requests in the chain are authenticated. The following technique based on hash chains can realize this idea: Before a round of diagnosis, the victim uses a publicly known one-way hash function $\mathcal{H}()$ to generate a hash chain: $n_h = \mathcal{H}(n_{h+1})$, where $h = 0, \ldots, 2^a - 1$ and n_{2^a-1} is a nonce, i.e., a secret bit sequence that changes for each round of diagnosis. A DII request issued to a router at hop h, say R_x, needs to contain n_h. The subsequent DII request that the victim sends to one of R_x's upstream routers, say R_y, has to carry n_{h+1} to be accepted by R_y. Router R_y can check the validity of n_{h+1} by checking whether $n_h = \mathcal{H}(n_{h+1})$, since R_x piggybacked n_h in a DAI request sent to R_y earlier.[5] Due to the property of one-way hash functions, an attacker that has eavesdropped on the value of n_h cannot compute n_{h+1} easily. Therefore, as long as n_{h+1} is long enough to ensure that the probability of guessing its value within the time period between two consecutive DII requests is negligible, hash chains should be sufficient to authenticate these requests. A hash chain is regenerated for each round of diagnosis to counter replay attacks. The advantage of the proposed technique is that it does not require the existence of a key distribution infrastructure.

Besides DAI and DII forgeries, other attacks are possible. For instance, an attacker may attempt to forge information in the marking fields of packets. Fortunately, such attacks—although they may be effective against probabilistic packet marking schemes (e.g., [7], [11], [28], [31], [42])—are not effective against AD or PAD. Since AD and PAD use deterministic packet marking, a forged marking will be overwritten by intermediate routers. In a different type of attack, an adversary may attempt to circumvent AD or PAD by enabling the attackers to send packets intermittently so that a diagnosis process would halt at some intermediate hop. AD or PAD is not designed to handle such attacks. However, we note that a possible solution is to enable the victim to detect such intermittent "pulsing" or "shrew" attacks using DWT (*Discrete Wavelet Transform*) [19] or DFT (*Discrete Fourier Transform*) [4]. Then, AD or PAD can be used in the same way as described in Section 2.

3.3 Issues of Router and Network Overhead

Although the packet-marking procedure required by AD/PAD adds additional router overhead, it is well within the capability of conventional routers. For instance, *input debugging* [32]—the functionality that can determine the interface that processed a particular packet—is widely supported by today's routers. Furthermore, the routine tasks of looking up routing tables and updating packets' *TTL* and *Checksum* fields are not much different from the required operations of the proposed marking procedure. More importantly, since AD and PAD are reactive defense mechanisms, it is unlikely that a router will receive a large number of simultaneous requests for packet marking. This ensures that a router will not be overburdened with packet marking tasks the vast majority of the time.

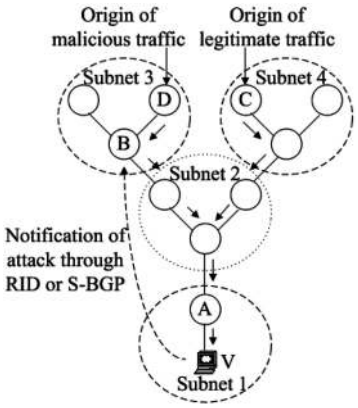

Fig. 3. A gradual deployment scenario. (Subnets 1, 3, and 4 support AD/PAD while Subnet 2 does not.).

Another concern is about the impact of network congestion caused by DDoS attacks on the timely transmission of AD/PAD commands. There are two aspects that can help mitigate this impact. First, the commands the victim sends are from downstream to upstream, while the flooding goes the reverse direction. In a duplex link, which is the case for most core networks, traffic in one direction is not affected by that in the opposite direction. Secondly, AD commands could contain some rate-limit requests (as Pushback [20] does), so that, when an upstream router receives the command, it will rate-limit the traffic to the victim and not cause congestion near the victim. Thus, the status packet from upstream to downstream should not be affected by the attack. In addition, another straightforward solution is to prioritize authenticated AD commands.

3.4 Gradual Deployment Considerations

Since the instantaneous wide deployment of a new DDoS countermeasure is not possible, considerations for gradual deployment must be given. AD and PAD can support gradual deployment at the subnet[6] level in two ways. In the first approach, AD/PAD is implemented only at the victim's neighbor subnets, forming a perimeter of defense around the victim (which we refer to as the "perimeter approach" hereafter). The distance between the outer boundary of the perimeter and the victim has to be large enough so that the attack paths are sufficiently diverged at the boundary. Note that attack paths converge as they get closer to the victim. If the distance is sufficiently large, AD/PAD is expected to throttle attacks without causing considerable collateral damage to legitimate traffic. The other approach relies on the distributed attack detection (which we refer to as the "distributed diagnosis approach" hereafter). In this approach, the border routers of subnets, which support AD and PAD, coordinate the diagnosis process under the help of attack detection devices. An example is shown in Fig. 3. When a victim detects an attack,

5. Note that this nonce can also be used to authenticate the status packet sent by R_y to the victim.

6. A subnet is defined as an Autonomous System (AS) or an area of an AS that is administrated by a single ISP.

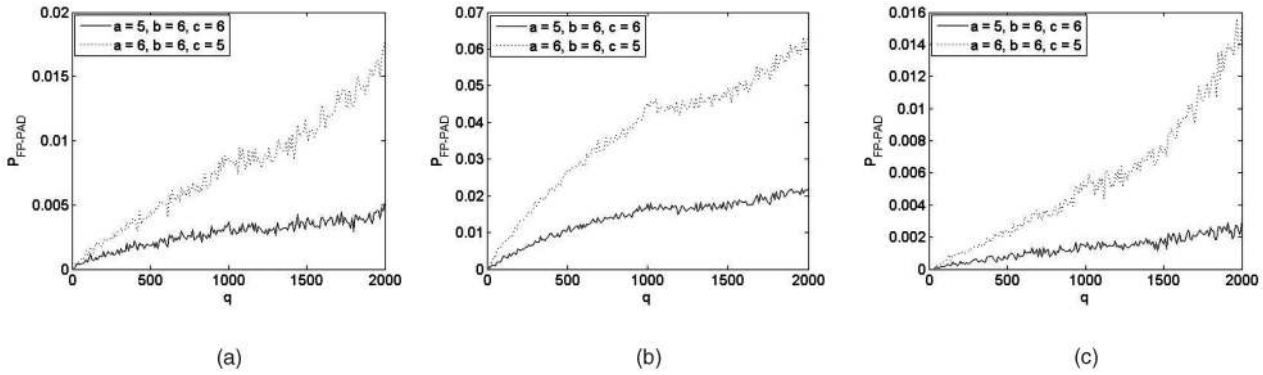

Fig. 4. P_{FP-PAD} versus q in (a) the Skitter map, (b) the Lumeta's map, and (c) the 6-degree complete tree topology.

it instructs the upstream subnets to initiate the diagnosis process by utilizing secure signaling techniques such as the Real-time Internetwork Defense (RID) [25] or S-BGP [15]. In response, the border routers of the subnets that support AD/PAD initiate the diagnosis process, limiting the process within the subnet. Suppose that Subnet 2 in Fig. 3 does not support AD/PAD and Subnets 1, 3, and 4 do. Also, suppose that attack traffic is coming from router D and legitimate traffic is coming from router C. If the perimeter approach is used, then all legitimate traffic passing through Subnet 2 and destined for V will be throttled by router A. The collateral damage can be significantly reduced with the distributed diagnosis approach. In the distributed diagnosis approach, A would instruct B to execute a diagnosis process within Subnet 3. If B is installed with a device that can detect attack packets destined for V, B can run a diagnosis on behalf of V according to the procedure described in Section 2. Then, B would instruct the upstream router D to throttle the attack traffic. Router A would restrain from filtering since there would be no attack traffic arriving at A when it is throttled at D. As a result, C's traffic is no longer throttled. In Section 4, we investigate the false positive ratios of AD under different deployment scenarios using simulations.

4 Simulation and Results

We adopt three different network topologies for our simulations. The first topology is chosen from the Skitter Internet map [29]. Based on the link data collected on 1 January 2006, we first chose a router with a degree of six as the victim's edge router and then randomly chose multiple distinct routes originating from it. The second topology is selected from the network topology data of Lumeta's Internet Mapping Project [3]. The provided path data was collected on 8 February 2006. All paths start from a single router, which is the victim's edge router in our simulation. This edge router has a degree of two. The third network topology is an n-degree complete tree model. In the n-degree complete tree model, we model the victim's upstream routers as a subset of a full and complete tree rooted by the victim's edge router, with every router having n connected routers or hosts (i.e., one parent router and $(n-1)$ children routers or hosts). The average router degree of the traced routes studied in the Skitter project was found to be 6.34 [29]. Therefore in our simulations, we employ $n = 6$.

According to the experimental findings of [13], a typical distribution of the number of hops from clients to a server can be regarded as a Gaussian distribution with a mean of 16.5 and a standard deviation of 4. In our simulation experiments, we use the same distribution for generating the routes' number of hops for the network topologies based on the Skitter map and the 6-degree complete tree topology model. The upper and lower bounds of the hop counts are set to 32 and 1, respectively. Generation of hop count is not required for Lumeta's map because it already contains hop count information.

4.1 P_{FP-PAD} versus q

Figs. 4a, 4b, and 4c show PAD's false positive ratios[7] versus the number of attackers diagnosed per round in the three network topologies described previously. Two settings of the marking field lengths, $a = 5$, $b = 6$, and $c = 6$ (referred to as Setting 1 hereafter) and $a = 6$, $b = 6$, and $c = 5$ (referred to as Setting 2 hereafter), were simulated for each topology. In these simulations, we fixed the number of attackers, N_a, at 2,000 and the total number of clients at 5,000. We varied q from 10 to 2,000 in increments of 10. Each datum is the average of 10 independent experiments.

From these results, it can be seen that P_{FP-PAD} is an increasing function of q. The false positive ratio, P_{FP-PAD}, had the largest value when $q = 2{,}000$ under Setting 2 in the Lumeta map. The big difference between the two settings is due to the different value of $b + c$, as (2) indicates. We found that the value of a changing from five to six makes little difference because, in our simulation, different routes diverge within 32 hops at an overwhelming probability, even when enhanced packet marking procedure is used. Therefore, we favor Setting 1 and employ it for all simulations from Sections 4.2, 4.3, and 4.4.

We also noticed that the Lumeta map has the highest P_{FP-PAD} compared to the other two topologies. The difference is caused by the different degrees of the victim's edge routers. In the Skitter map and the 6-degree complete tree topology, the victim's edge router has a degree of six, while, in Lumeta's map, it is two. Consequently, the routes in Lumeta's map, compared to those in the other two topologies, share more common links and induce a smaller value of m in (3) and, thus, a greater P_{FP-PAD} value. This result shows that making a victim network multihomed can decrease PAD's false positive ratio.

7. We use the term "false positive ratio" to denote the observed false positive probability.

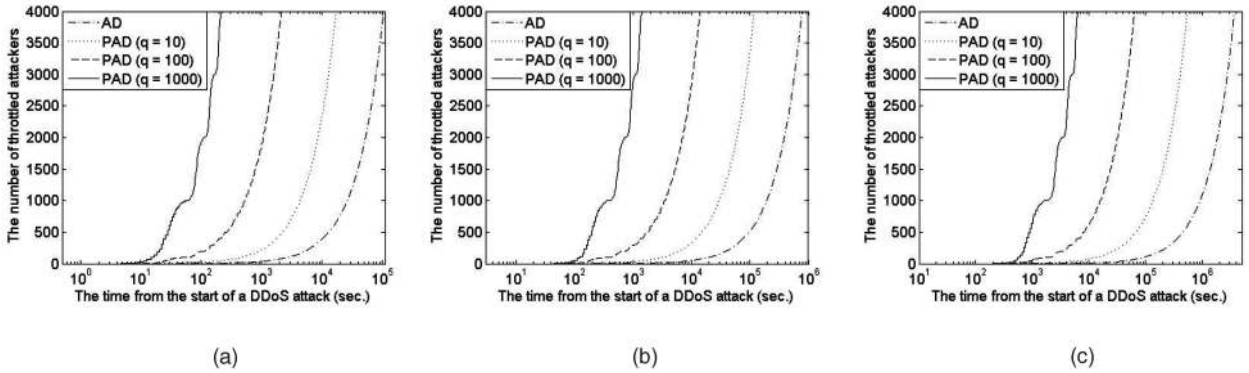

Fig. 5. Number of throttled attackers over time under (a) Setting A, (b) Setting B, and (c) Setting C.

4.2 Attack Mitigation Delay

To simulate the attack mitigation delay, T (which corresponds to both T_{AD} and T_{PAD} in Section 2.6), we used an actual network delay distribution measured from the Internet [10]. We assumed $Q = 1{,}000$ (packets) and used three settings for the values of $T_{pro}(i,j)$ and $R(i,j)$.

- Setting A:

$$T_{pro}(i,j) = 0.1 \text{ sec}, \ R(i,j) = 1{,}000 \text{ packets/sec } (\forall i,j).$$

- Setting B:

$$T_{pro}(i,j) = 5 \text{ sec}, \ R(i,j) = 1{,}000 \text{ packets/sec } (\forall i,j).$$

- Setting C:

$$T_{pro}(i,j) = 0.1 \text{ sec}, \ R(i,j) = 20 \text{ packets/sec } (\forall i,j).$$

Figs. 5a, 5b, and 5c show the plots of the number of throttled attackers versus time for AD and PAD under the three settings of $T_{pro}(i,j)$ and $R(i,j)$. All simulations were run on the Skitter map topology. The results show that T is roughly inverse proportional to q (q can be regarded as one in AD). This is expected since handling q attackers in parallel should accelerate the diagnosis process by approximately q times. From the difference between Fig. 5a and Fig. 5b and that between Fig. 5a and Fig. 5c, it can be inferred that $T_{pro}(i,j)$ has less impact on T than $R(i,j)$ does.

Fig. 6 shows T versus q when 4,000 out of 5,000 clients are attackers in the Skitter map topology. The figure shows the inverse proportional relationship between T and q.

4.3 The Trade-Off between False Positives and Attack Mitigation Delay

Previous simulation results have shown that, when q increases, P_{PF-PAD} increases while T decreases. Since it is desirable to minimize both P_{PF-PAD} and T, a trade-off between the two parameters exists. Fig. 7 shows a plot of P_{FP-PAD} versus T. One can readily observe their trade-off relationship. An appropriate trade-off can be made by choosing an appropriate value of q.

4.4 Partial Subnet Deployment

In Section 3.4, we have discussed the two approaches to support the gradual deployment of AD and PAD. We have simulated them in Lumeta's map topology to observe the false positive ratio under different deployment scenarios. Only AD is used in these simulations. To describe the simulation environment for the perimeter approach, we define the parameter, dd, called deployment depth, that represents the hop count in the unit of AD-capable subnets that are located immediately upstream of a victim. Fig. 8 shows an illustrative example using the topology of Lumeta's map. If $dd = 2$, then subnet 65.198.68.0/24, subnet 157.130.0.0/16, and subnet 152.63.0.0/16 support AD. If $dd = 3$, then these three subnets together with the 464 subnets connected to subnet 152.63.0.0/16 all support AD. Fig. 9a shows the false positive ratio when dd varies from 2 to 7. The number of attackers is varied from 1 to 2,001 in increments of 10 and each datum is the average of three independent simulations. The total number of clients is 5,000.

In another set of simulations, we assume that the distributed diagnosis approach for supporting gradual deployment is employed (see Section 3.4 for details). We

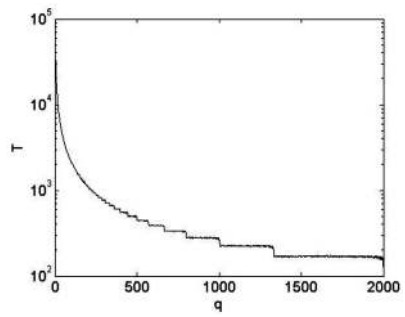

Fig. 6. T versus q.

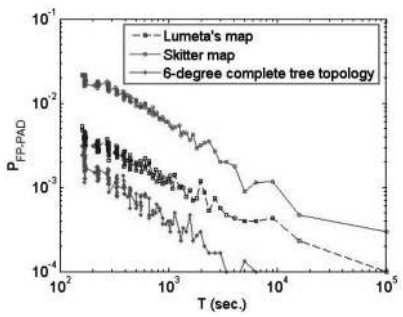

Fig. 7. Trade-off between P_{FP-PAD} and T.

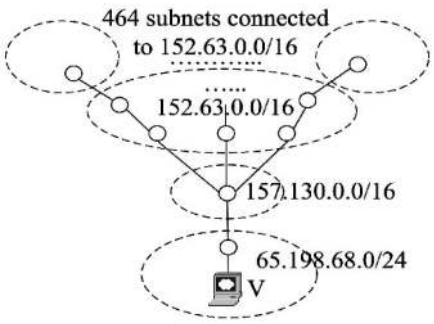

Fig. 8. A partial deployment scenario.

assume that subnets 65.198.68.0/24, 157.130.0.0/16, and 152.63.0.0/16 have implemented AD and that the subnets supporting distributed diagnosis approach are distributed uniformly throughout the Internet. We vary the deployment percentage from 20 percent to 80 percent in increments of 20 percent, and the simulation results are shown in Fig. 9b. Through our simulation experiments, we are able to observe that implementing AD in subnet 157.130.0.0/16 and subnet 152.63.0.0/16 is crucial for lowering the false positive ratio. Fig. 9c shows the false positive ratios when the subnets at deployment depths one and two do not support AD. In this case, the false positive ratios are unacceptably high for all partial deployment scenarios. This fact leads to the conclusion that the subnets located immediately upstream of a victim must support AD in order for AD to be a viable DDoS countermeasure. Since we have assumed having no attack signature and tolerating no false negatives, if the subnets immediately upstream of the victim's subnet do not support AD, then it is likely that some attack packets will arrive at the AD-capable edge router in a victim's subnet, which will filter all packets, whether legitimate or malicious, destined for the victim. It is obvious that filtering packets so close to the victim based solely on the destination address will cause the false positive ratio to increase to an unacceptable level.

5 Related Work

Many DDoS attack mitigation schemes have been proposed in the literature. Several techniques try to solve the problem from the perspective of fair resource allocation. One technique that belongs to this category is client puzzles [9], [14], [21], [35], which force a client to provide the solution to a cryptographic "puzzle" before any resource is committed. A client has to consume some of its own resources to compute the solution, and the number of resources that a client needs to commit is commensurate with the puzzle difficulty. However, client puzzles are criticized for inducing high request overhead and weak service-access guarantees [12]. As an improved alternative, a *ticket-based* scheme was proposed in [12]. In this scheme, a client needs to obtain a ticket from a protected server, which generates tickets at line speeds, to access its service. A similar idea is to enforce a client to get the *capability* from the server it is accessing [41], [43]. Routers will block transmissions lacking capability. Another technique for fair resource allocation is max-min servercentric router throttles [38]. This scheme enables a server under attack to contact a perimeter of upstream routers to install *router throttles*, where each router only forwards the max-min fair share of the traffic that is allowed by the server. In [36], a different approach that utilizes Differentiated Service (DiffServ) proposes to isolate TCP, UDP, and ICMP traffic, and to bound the resource consumption of UDP and ICMP traffic. The bidirectional traffic of TCP connections and TCP control segments are also differentiated.

Another category of attack mitigation schemes utilize an attack detection model to differentiate attack traffic from legitimate traffic and use a packet filtering module to filter attack packets. The schemes presented in [22], [26], [33], [34], [37], [39], [40] all place the two modules close to each other. According to the location of the modules, these schemes can be to victim-based, network-based, or source-based. Schemes proposed in [33], [34], [37], [39], [40] are victim-based. For instance, the filtering schemes in [33], [39], [40] rely on packet marking to filter attack traffic—each router marks packets and the markings of the received packets are used by the victim to distinguish attack packets from legitimate packets. A typical network-based scheme is proposed by Park and Lee in [26]. They suggest installing packet filters in AS border routers to prevent IP spoofing. D-WARD [22] is a scheme representative of being source-based. It throttles attackers at the source by suppressing any host which sends a much heavier volume of traffic than it receives.

The schemes proposed in [6], [20], [27], [44] place the attack detection module near the victim and execute packet filtering close to the attack sources. As a representative scheme, Pushback [20] adopts hop-by-hop transmission of control messages to coordinate packet filtering. The

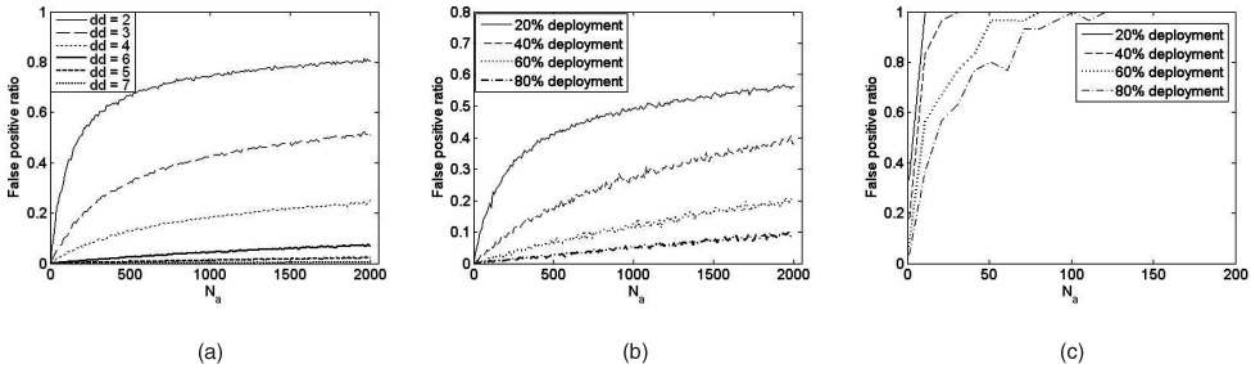

Fig. 9. The false positive ratios of AD with gradual deployment: (a) for various deployment depths, (b) $dd = 2$ and uniformly random deployment for all other subnets in the Internet, and (c) the victim's edge router supports AD, subnets at deployment depths one and two from the victim do not support AD, and uniformly random deployment for all other subnets in the Internet.

authenticity of control message packets is ensured using the *TTL* field. This scheme uses the *aggregate-based congestion control* (ACC) module for local congestion detection and high-bandwidth traffic throttling. If a router cannot adequately throttle an aggregate by itself, the Pushback module requests the router's upstream routers to rate-limit the aggregate together.

Schemes in [7], [11], [17], [28], [31], [42] utilize probabilistic packet marking for IP traceback. These schemes overload the 16-bit *Identification* field in the IP header to mark partial path information in each packet. After many packets are received, the complete information can be recovered from the markings of these packets. The major drawback of these schemes is their vulnerability to the packet-markings forgery [42]. Another type of IP traceback approach employs the technique of packet logging [18], [30]. Such schemes are still not practical since they do not support gradual deployment and induce too much overhead on routers [42].

6 Conclusion

In this paper, we proposed two novel DDoS countermeasures, AD and PAD. They employ a "divide-and-conquer" strategy to isolate attacking hosts and filter their traffic. AD/PAD integrates the concepts of Pushback and packet marking. AD/PAD's framework is in line with the ideal framework of DDoS mitigation schemes in which the attack detection module is placed at the victim end and the filtering module is placed close to the attack sources [2], [23]. Recognizing AD's inability to handle large-scale attacks, we introduced its parallelized version, called PAD. PAD is capable of tracing back and mitigating attack traffic from multiple attackers simultaneously, thus enabling it to handle large-scale attacks. Our analysis and simulation results indicate that AD/PAD has several advantageous features, including

1. is reactive so as to incur limited overhead,
2. does not rely on attack signatures for filtering attack traffic,
3. is robust against IP spoofing and marking-field forgeries,
4. supports incremental deployment, and
5. incurs low false positives.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. A preliminary version of portions of this material was presented in [5].

References

[1] B. Al-Duwairi and T.E. Daniels, "Topology Based Packet Marking," *Proc. IEEE Int'l Conf. Computer Comm. and Networks (ICCCN),* pp. 146-151, Oct. 2004.

[2] R.K.C. Chang, "Defending against Flooding-Based Distributed Denial-of-Service Attacks: A Tutorial," *IEEE Comm. Magazine,* vol. 40, no. 10, pp. 42-51, Oct. 2002.

[3] H. Cheswicka and H. Burch, *Internet Mapping Project,* http://research.lumeta.com/ches/map/, 2006.

[4] Y. Chen, K. Hwang, and Y.-K. Kwok, "Collaborative Defense against Periodic Shrew DDoS Attacks in Frequency Domain," http://gridsec.usc.edu/files/TR/ACMTISSEC-LowRateAttack-May3-05.pdf, 2005.

[5] R. Chen and J.-M. Park, "Attack Diagnosis: Throttling Distributed Denial-of-Service Attacks Close to the Attack Sources," *Proc. IEEE Int'l Conf. Computer Comm. and Networks (ICCCN),* pp. 275-280, Oct. 2005.

[6] S. Chen and Q. Song, "Perimeter-Based Defense against High Bandwidth DDoS Attacks," *IEEE Trans. Parallel and Distributed Systems,* vol. 16, no. 6, pp. 526-537, June 2005.

[7] D. Dean, M. Franklin, and A. Stubblefield, "An Algebraic Approach to IP Traceback," *ACM Trans. Information and System Security (TISSEC),* vol. 5, no. 2, pp. 119-137, May 2002.

[8] C. Douligeris and A. Mitrokotsa, "DDoS Attacks and Defense Mechanisms: Classification and State-of-the-Art," *Computer Networks,* vol. 44, no. 5, pp. 643-666, Apr. 2004.

[9] W. Feng, E. Kaiser, and A. Luu, "Design and Implementation of Network Puzzles," *Proc. IEEE INFOCOM,* pp. 2372-2382, Mar. 2005.

[10] A. Fei, G. Pei, R. Liu, and L. Zhang, "Measurements on Delay and Hop-Count of the Internet," http://irl.cs.ucla.edu/papers/internet-measurement.ps.gz., 2005.

[11] M.T. Goodrich, "Efficient Packet Marking for Large Scale IP Traceback," *Proc. ACM Conf. Computer and Comm. Security (CCS),* pp. 117-126, Nov. 2002.

[12] V.D. Gligor, "Guaranteeing Access in Spite of Distributed Service-Flooding Attacks," *Proc. Security Protocols Workshop,* pp. 66-74, Apr. 2003.

[13] C. Jin, H. Wang, and K.G. Shin, "Hop-Count Filtering: An Effective Defense against Spoofed DoS Traffic," *Proc. ACM Conf. Computer and Comm. Security (CCS),* pp. 30-41, Oct. 2003.

[14] A. Juels and J. Brainard, "Client Puzzle: A Cryptographic Defense against Connection Depletion Attacks," *Proc. Network and Distributed System Security Symp.,* pp. 151-165, Feb. 1999.

[15] S. Kent, "Securing the Border Gateway Protocol: A Status Update," http://www.ir.bbn.com/sbgp/S-BGP_CMS2003-Kent.pdf, 2005.

[16] T. Killalea, *Recommended Internet Service Provider Security Services and Procedures,* IETF RFC 3013, Nov. 2000.

[17] T.K.T. Law, J.C.S. Lui, and D.K.Y. Yau, "You Can Run, but You Can't Hide: An Effective Statistical Methodology to Trace Back DDoS Attackers," *IEEE Trans. Parallel and Distributed Systems,* vol. 16, no. 9, pp. 799-813, Sept. 2005.

[18] J. Li, M. Sung, J. Xu, and L. Li, "Large-Scale IP Traceback in High-Speed Internet: Practical Techniques and Theoretical Foundation," *Proc. IEEE Symp. Security and Privacy,* pp. 115-129, May 2004.

[19] X. Luo and R.K.C. Chang, "On a New Class of Pulsing Denial-of-Service Attacks and the Defense," *Proc. Network and Distributed System Security Symp.,* Feb. 2005.

[20] R. Mahajan, S.M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling High Bandwidth Aggregates in the Network," *Computer Comm. Rev.,* vol. 32, no. 3, pp. 62-73, July 2002.

[21] T. McNevin, J.-M. Park, and R. Marchany, "Chained Puzzles: A Novel Framework for IP-Layer Client Puzzles," *Proc. IEEE Int'l Conf. Wireless Networks, Comm., and Mobile Computing (WirelessCom '05),* June 2005.

[22] J. Mirkovic, G. Prier, and P. Reiher, "Attacking DDoS at the Source," *Proc. IEEE Int'l Conf. Network Protocols (ICNP),* pp. 312-321, Nov. 2002.

[23] J. Mirkovic, M. Robinson, and P. Reiher, "Alliance Formation for DDoS Defense," *Proc. Workshop New Security Paradigms,* pp. 11-18, Aug. 2003.

[24] J. Mirkovic and P. Reiher, "A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms," *ACM SIGCOMM Computer Comm. Rev.,* vol. 34, no. 2, pp. 39-54, Apr. 2004.

[25] K.M. Moriarty, "Distributed Denial of Service Incident Handling: Real-Time Inter-Network Defense," IETF Internet draft, work in progress, Feb. 2004.

[26] K. Park and H. Lee, "On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets," *Proc. ACM SIGCOMM,* pp. 15-26, Aug. 2001.

[27] C. Papadopoulos, R. Lindell, J. Mehringer, A. Hussain, and R. Govindan, "Cossack: Coordinated Suppression of Simultaneous Attacks," *Proc. DARPA Information Survivability Conf. and Exposition,* vol. 1, pp. 2-13, Apr. 2003.

[28] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical Network Support for IP Traceback," *Proc. ACM SIGCOMM,* pp. 295-306, Aug. 2000.

[29] "Skitter," http://www.caida.org/tools/measurement/skitter/index.xml, Cooperative Assoc. for Internet Data Analysis, 2006.

[30] A. Snoeren, C. Partridge, L. Sanchez, C. Jones, F. Tchakountio, S. Kent, and W. Strayer, "Hash-Based IP Traceback," *Proc. ACM SIGCOMM,* pp. 3-14, Aug. 2001.
[31] D.X. Song and A. Perrig, "Advanced and Authenticated Marking Schemes for IP Traceback," *Proc. IEEE INFOCOM,* pp. 878-886, Apr. 2001.
[32] R. Stone, "CenterTrack: An IP Overlay Network for Tracking DoS Floods," *Proc. USENIX Security Symp.,* pp. 199-212, July 2000.
[33] M. Sung and J. Xu, "IP Traceback-Based Intelligent Packet Filtering: A Novel Technique for Defending against Internet DDoS Attacks," *IEEE Trans. Parallel and Distributed Systems,* vol. 14, no. 9, pp. 861-872, Sept. 2003.
[34] R. Thomas, B. Mark, T. Johnson, and J. Croall, "NetBouncer: Client-Legitimacy-Based High-Performance DDoS Filtering," *Proc. DARPA Information Survivability Conf. and Exposition,* vol. 1, pp. 14-25, Apr. 2003.
[35] X. Wang and M.K. Reiter, "Mitigating Bandwidth-Exhaustion Attacks Using Congestion Puzzles," *Proc. ACM Conf. Computer and Comm. Security (CCS),* pp. 257-267, Oct. 2004.
[36] H. Wang and K.G. Shin, "Transport-Aware IP Routers: A Built-In Protection Mechanism to Counter DDoS Attacks," *IEEE Trans. Parallel and Distributed Systems,* vol. 14, no. 9, pp. 873-884, Sept. 2003.
[37] J. Xu and W. Lee, "Sustaining Availability of Web Services under Distributed Denial of Service Attacks," *IEEE Trans. Computers,* vol. 52, no. 2, pp. 195-208, Feb. 2003.
[38] D.K.Y. Yau, J.C.S. Lui, F. Liang, and Y. Yam, "Defending against Distributed Denial-of-Service Attacks with Max-Min Fair Server-Centric Router Throttles," *IEEE/ACM Trans. Networking,* pp. 29-42, Feb. 2005.
[39] A. Yaar, A. Perrig, and D. Song, "Pi: A Path Identification Mechanism to Defend against DDoS Attacks," *Proc. IEEE Symp. Security and Privacy,* pp. 93-107, May 2003.
[40] A. Yaar, A. Perrig, and D. Song, "StackPi: A New Defensive Mechanism against IP Spoofing and DDoS Attacks," Technical Report CMU-CS-02-208, Carnegie Mellon Univ., Feb. 2003.
[41] A. Yaar, A. Perrig, and D. Song, "SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks," *Proc. IEEE Symp. Security and Privacy,* pp. 130-143, May 2004.
[42] A. Yaar, A. Perrig, and D. Song, "FIT: Fast Internet Traceback," *Proc. IEEE INFOCOM,* pp. 1395-1406, Mar. 2005.
[43] X. Yang, D. Wetherall, and T. Anderson, "A DoS-Limiting Network Architecture," *Proc. ACM SIGCOMM,* pp. 241-252, Aug. 2005.
[44] S. Zhang and P. Dasgupta, "Denying Denial of Service Attacks: A Router Based Solution," *Proc. Int'l Conf. Internet Computing,* June 2003.

Ruiliang Chen received the bachelor's degree in communications engineering in 2000 and the master's degree in communications and information systems in 2003, both from Fudan University, China. He is currently a PhD student in the Bradley Department of Electrical and Computer Engineering at Virginia Polytechnic Institute and State University (Virginia Tech). From June 2003 to July 2004, he worked as a product engineer at the Intel Shanghai Product Corporation. His research interests include traceback and mitigation mechanisms for thwarting denial-of-service attacks, attack-resilient routing protocols for wireless ad hoc networks, and security issues in cognitive radio networks. He is a student member of the IEEE.

Jung-Min Park received the bachelor's and master's degrees in electronic engineering from Yonsei University, Seoul, Republic of Korea, in 1995 and 1997, respectively, and the PhD degree from the School of Electrical and Computer Engineering at Purdue University in 2003. From 1997 to 1998, he was a cellular systems engineer at Motorola Korea, Inc. In the fall of 2003, he joined the faculty of the Bradley Department of Electrical and Computer Engineering at Virginia Polytechnic Institute and State University (Virginia Tech) as an assistant professor. He is a recipient of the 1998 AT&T Leadership Award. Dr. Park's research interests include DoS attack countermeasures, e-commerce protocols, cognitive radio networks, key management, and applied cryptography. He is a member of the IEEE and the ACM.

Randolph Marchany has been involved in the computer industry since 1972. He is currently the director of the Virginia Polytechnic Institute and State University (Virginia Tech) IT Security Testing Lab, a component of the university's Information Technology Security Office. He is the coordinator of VA-CIRT, an incident response team comprised of IRTs from various Viriginia state universities. He is a coauthor of the FBI/SANS Institute's "Top 10/20 Internet Security Vulnerabilities" document that has become a standard for most computer security and auditing software and a co-author of the SANS Institute's "Responding to Distributed Denial of Service Attacks" document that was prepared at the request of the White House in response to the DDOS attacks of 2000. He is a coauthor of the SANS Institute's "Computer Security—Incident Handling—Step by Step," which has been recognized as one of the foremost publications on incident response. He has been a member of the SANS Institute's faculty since 1992 and is one of the developers of their GIAC security certification courses. He is a member of the EDUCAUSE security task force. He was a member of the White House Partnership for Critical Infrastructure Security working group that developed the Consensus Roadmap for responding to the recent series of DDOS Internet Attacks. He was a recipient of the SANS Institute's Security Technology Leadership Award for 2000, the Virigina Governor's Technology Silver Award in 2003, and an EDUCAUSE award in 2005. He is a member of the IEEE.