Pi: A Path Identification Mechanism to Defend against DDoS Attacks

Abraham Yaar Adrian Perrig Dawn Song

Carnegie Mellon University

{ayaar, perrig, dawnsong}@cmu.edu

Abstract

Distributed Denial of Service (DDoS) attacks continue to plague the Internet. Defense against these attacks is complicated by spoofed source IP addresses, which make it difficult to determine a packet's true origin. We propose Pi (short for Path Identifier), a new packet marking approach in which a path fingerprint is embedded in each packet, enabling a victim to identify packets traversing the same paths through the Internet on a per packet basis, regardless of source IP address spoofing.

Pi features many unique properties. It is a per-packet deterministic mechanism: each packet traveling along the same path carries the same identifier. This allows the victim to take a proactive role in defending against a DDoS attack by using the Pi mark to filter out packets matching the attackers' identifiers on a per packet basis. The Pi scheme performs well under large-scale DDoS attacks consisting of thousands of attackers, and is effective even when only half the routers in the Internet participate in packet marking. Pi marking and filtering are both extremely light-weight and require negligible state.

We use traceroute maps of real Internet topologies (e.g., CAIDA's Skitter [6] and Burch and Cheswick's Internet Map [4, 14]) to simulate DDoS attacks and validate our design.

Keywords: *Denial-of-service, DoS, Internet, DoS defense, packet marking, path identifier.*

1 Introduction

Distributed denial of service (DDoS) attacks continue to plague the Internet. In a typical DDoS attack, attackers compromise multiple machines and use them to send large numbers of packets to a single victim server to overwhelm its capacity. For example, on October 21, 2002, an attacker flooded the root DNS servers with traffic in an effort to deprive the Internet of the DNS name lookup service (which would have paralyzed the majority of Internet applications). Only five out of thirteen root servers were able to withstand the attack [23]. Previously, DDoS attacks had shut down several large Internet sites, such as Yahoo! and eBay.

As an increasing number of businesses and services depend on the Internet, safeguarding them against attacks is a priority. Some critical infrastructures — for example, emergency telephone response (911) — increasingly rely on the Internet for communication and coordination [2]. Clearly, critical services demand effective countermeasures against DDoS attacks.

One challenge in defending against DDoS attacks is that attackers often use spoofed source IP addresses (hereafter referred to as *spoofed IP addresses*) which make it difficult to identify and block their packets under the current Internet infrastructure. Because of the importance and urgency of the DDoS problem, many researchers have studied countermeasures (we review their efforts in Section 8). A common solution in proposed systems is a *traceback* mechanism that has routers mark information on packets en-route to the victim, who can then use that information to reconstruct the path that the packets take from the attacker through the Internet, despite IP address spoofing. The path information obtained by the traceback mechanism can then be used to install network filters upstream from the victim to block attack traffic. The common assumption in these mechanisms is the need to reconstruct the exact path (or a path prefix) to the attacker in order to defend the victim. Most of these mechanisms (with the exception of [36]) also assume that the victim only initiates the traceback or passively receives traceback information, but does not otherwise actively participate in packet filtering. These assumptions face the following shortcomings:

- The victim must receive large numbers of packets before it is able to reconstruct the path that they are taking.
- Routers and/or victims need to perform non-trivial operations in marking packets or in reconstructing paths.
- Network filtering is done on a per-flow or per-network basis using coarse identification criteria, rather than on

a per-packet basis.

- The victim has to rely on upstream routers to perform packet filtering, even once the attack paths have been identified.

In this paper, we present a new approach for defending against DDoS attacks that does not rely on these assumptions. We observe that reconstructing the exact path to the attacker is not necessary in defending against a DDoS attack — we only need to get an indication of the particular path that attack packets take. In addition, because our approach transmits path information in each packet, the victim can filter packets itself, based on its knowledge of the path information carried by a single prior attack packet.

Our approach embeds in each packet an identifier based on the router path that a packet traverses. The victim need only classify a single packet as malicious to be able to filter out all subsequent packets with the same marking. What makes this possible is that our packet marking is deterministic — all packets traversing the same path carry the same marking. All previous marking schemes that we are aware of are probabilistic in nature, in which the victim needs to collect a large number of packets to reconstruct the path. In our approach, a path identifier fits within a single packet so the victim can immediately filter traffic after receiving just one attack packet. Our scheme is extremely light-weight, both on the routers for marking, and on the victims for decoding and filtering. The router marking in our scheme is also robust to the presence of legacy routers and shows strong incremental deployment properties. Finally, our scheme can also be used to enhance the effectiveness of other DDoS countermeasures, for example, the Pushback framework [16, 21], as we discuss in Section 7.2.

The remainder of the paper is organized as follows: in Section 2 we classify different DDoS attacks and provide some assumptions that we use to help explain the Pi scheme. In Section 3 we provide a high-level overview of the Pi scheme and what makes it unique from previously proposed DDoS defense schemes. In Section 4 we present the packet marking algorithm that we propose to deploy on Internet routers. In Section 5 we discuss packet filters that use Pi marks to effectively filter attack traffic in DDoS attacks. In Section 6 we present a DDoS model and experiments showing the Pi scheme's performance under a DDoS attack. We discuss further applications and improvements to Pi in Section 7. In Section 8 we cover related work in the field of DDoS defense and monitoring, and Section 9 concludes the paper.

2 Problem Statement and Assumptions

Our proposal seeks to prevent DDoS attacks which use packet floods to consume network and server resources. We classify these attacks as follows:

- Network Resource Attack. In this attack, the attacker sends many useless packets to the victim server with the intention of depleting the network bandwidth connecting the server to the rest of the Internet. If this attack succeeds and network bandwidth is sufficiently depleted, legitimate users experience severe or complete service degradation because their packets are unable to reach the server.
- Server Resource Attacks.
  - Server Processing Attack. In this attack, similar to the Network Resource Attack, the attacker sends many useless packets with the intention of overwhelming the victim server's ability to process the increased load of packets. The server is then forced to drop incoming packets indiscriminately, and thus legitimate users experience service degradation or failure.
  - Server Memory Attack: In this attack, the attacker takes advantage of ambiguities in protocols to deplete the victim server's memory. Ambiguities can range from the reservation of resources for half-open connections (TCP SYN flooding attacks [7, 30]) to the buffering of packet fragments for a packet which the attacker will never completely send (IP fragmentation attacks). Researchers often propose computational solutions for this class of attacks [10, 12, 18]. We do not discuss these attacks further in this paper.

In dealing with the above attacks, we assume that the Pi filter, which uses Pi markings to make per-packet accept/drop decisions, can be deployed either on the victim's machine or, preferably, on a dedicated machine such as a modified firewall, and placed in the path to the victim server. In the case of a network resource attack, we assume that a Pi filter is deployed on the ISP's side of the last-hop link, and can thus filter packets before they consume the victim's network bandwidth.

Lastly, we assume that routers are capable of marking the IP Identification field of all packets that they forward, provided that the marking algorithm is reasonably small in both processor and memory usage. The use of the IP Identification field for packet marking is widespread in the literature, first proposed by Savage et al. [28, 29] and later in several other works on IP traceback [1, 31, 32].

3 Design Motivation

A common approach for DDoS defense is to provide the victim of a DDoS attack with the IP addresses of the routers along the path of the attack packets. With this information, the victim can request that upstream ISPs deploy packet filters to drop packets originating from the attacking networks, destined for the victim. Figure 1, shows that the average path length in the Internet, and thus, the average number of router IP addresses that must be transmitted to the victim, is roughly 15 (taken from Burch and Cheswick's Internet Mapping Project [4, 14] and from CAIDA's Skitter Map [6]). Assuming no compression, the lower bound on the amount of data the victim needs to reconstruct a single attack path of average length is 60 bytes. There are many proposed solutions for how to transmit this information to the victim; we review them in Section 8.

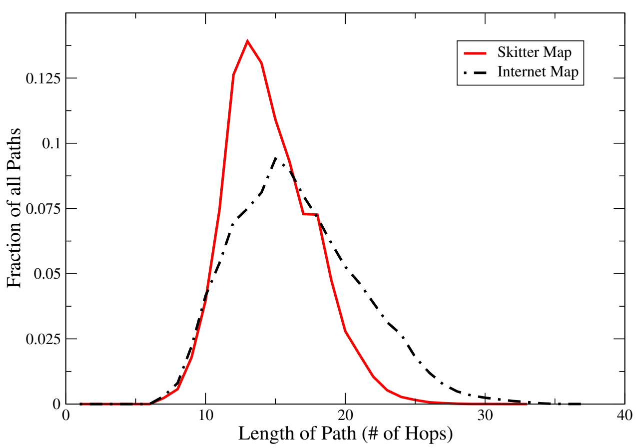


Figure 1. Distribution of Internet path lengths using the Skitter Map and the Internet Map.

We propose a new approach for dealing with the DDoS attack problem. Since DDoS attacks often involve compromised machines, co-opted by a group of hackers exploiting other security vulnerabilities, there is little incentive for the victim to identify the path to specific attacker machines other than the need to provide information to help upstream ISPs deploy packet filters as effectively — and with as little effect on legitimate traffic — as possible. However, if all packets arriving at the victim have some *distinctive marking*, then the victim need only note the markings that correspond to attack packets and then drop all incoming packets matching those markings.

To illustrate what we mean by the term *distinctive marking* we take the case of the Internet modeled as a complete-binary tree, rooted at the victim server, with n nodes at the leaves. Using the estimated current size of the Internet [15] as n, we get $\lceil \log_2(162, 128, 493) \rceil = 28$ bits to uniquely represent each path from the victim server to an end host (with a 0-bit representing a left branch and a 1-bit representing a right branch). Although this model is an exceedingly simple representation of the Internet, we use it to show that path information need not be exclusively constructed of router IP addresses. We propose to construct a *path identifier*, akin to the binary tree path representation, to be embedded by routers in the IP Identification field [1] of every packet they forward. The path identifier will act as the distinctive marking which the victim can use to filter incoming packets.

Unfotunately, the 28 bit path identifier in our binary tree example is still 12 bits larger than the 16 bits that are available in the IP Identification field. In the Pi scheme, we limit ourselves to 16 bit path identifiers so that each packet carries all the marking information of the scheme. The router markings in Pi are also determinisitic such that every packet traversing a particular path is marked with the same path identifier, which is generated piecemeal by the routers along the path from end-host to victim. Because each router has only local knowledge (last-hop, next-hop) of a particular path, the marking for an entire path in Pi is not guaranteed to be globally unique. However, we show that a globally unique identifier is not necessary in providing strong DDoS protection and that the benefits of having a single-packet, deterministic marking, allow the victim to develop rapidly responsive packet filters to protect itself during such attacks.

4 The Pi Marking Scheme

In this section, we present the Pi packet marking scheme to be deployed on Internet routers. We assume, for the moment, that all routers in the Internet implement our scheme, however, in Section 6.6 we show experimental results relating Pi's performance to the fraction of non-marking routers in the Internet.

4.1 Basic Pi Marking Scheme

In its simplest form, we propose an n-bit scheme where a router marks the last n bits of its IP address in the IP Identification field of the packets it forwards. To determine the location within the field to mark the bits, we break the field into $\lfloor 16/n \rfloor$ different marking sections, and use the value of the packet's TTL, modulo $\lfloor 16/n \rfloor$ as an index into the section of the field mark. Figure 2 shows the C code for the Pi basic marking scheme where the $markingbits$ function simply returns the IP address that is passed to it. Figure 3 shows an example marking scenario, using 1-bit marking

[1] Savage et al. [28, 29] were the first to argue that the IP Identification field is a good candidate for this use because the field is only used for packet fragmentation, which constitutes less than 0.25% of the packets in the Internet [34]. Our scheme can coexist with fragmentation if routers refrain from marking packet fragments. Although attackers may be tempted to circumvent our defense by sending only fragmented DDoS traffic, such an attack is readily blocked by existing packet filters that automatically drop packet fragments.

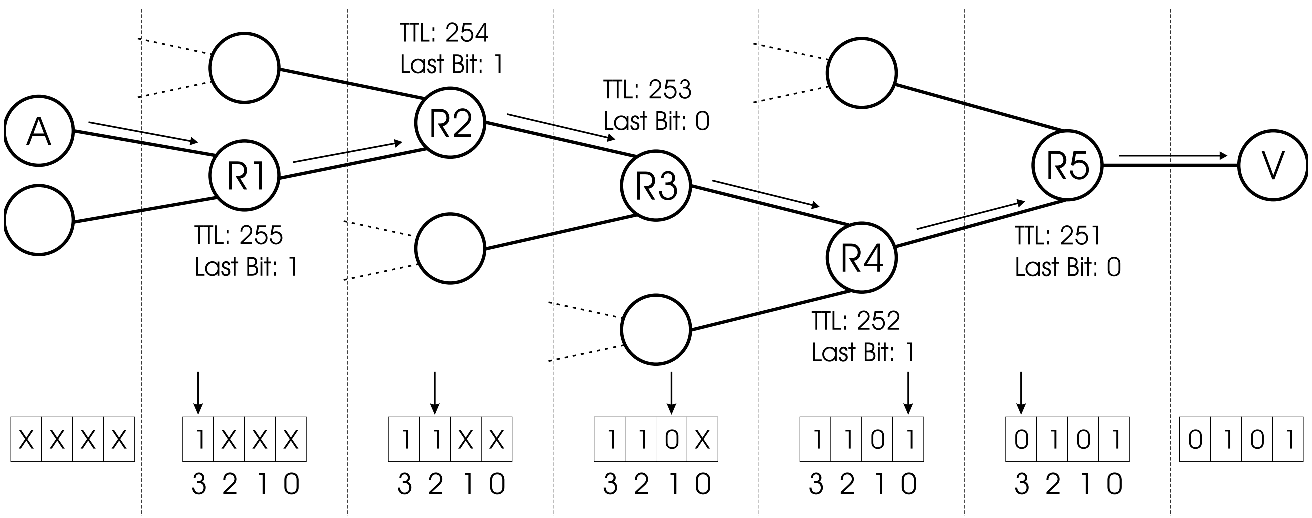


Figure 3. Example of our initial marking scheme. The packet travels from the attacker A to the victim V across the routers R1 to R5. Each router uses the TTL value of the packet to index into the IP identification field to insert its marking. In this example we show a 1-bit marking in a 4-bit field for simplicity.

$\mathcal{P}$ = Pi mark of the packet
n = number of bits each router marks

Pimark($\mathcal{P}$, *TTL*, $Curr\_IP$, n)
{
$m = 2^n - 1$;
$b = markingbits(Curr\_IP)\ \&\ m$;
$bitpos = (TTL \mod \lfloor \frac{16}{n} \rfloor)\, n$;
$b\ <<\ bitpos$;
$m\ <<\ bitpos$;
return(($\mathcal{P}$ & $\sim$*m*) | *b);*
}

Figure 2. The Pi Marking Algorithm

in a 4-bit field. In the remainder of this section, we discuss the design decisions and some improvements to the $markingbits$ function that greatly enhance the uniqueness of a particular Pi mark.

4.2 IP Address Hashing

We find that the distribution of the last bits of the IP addresses of the routers from our sample Internet data is highly skewed. This is problematic because if, for example, ISPs tended to designate router IP addresses with the last byte as 0, then many of our packet markings would be zero, which would make the Pi markings for different paths less likely to be distinguishable from each other. Ideally, we would like to maximize the entropy of the bits that we mark with, to reduce the likelihood of marking collisions (where two different paths have the same Pi marking). Table 1 shows the distribution of the last bits of the routers' IP addresses from the Internet Map.

Value	Bit 0	Bits 1 to 0	Bits 2 to 0
0	80170	11993	3004
1	51095	39755	24725
2		68177	40213
3		11340	8483
4			8989
5			15030
6			27964
7			2857
Variance	$422.7 \cdot 10^6$	$731.1 \cdot 10^6$	$178.9 \cdot 10^6$

Table 1. Distribution of the least significant 1 to 3 bits of the routers' IP addresses from the Internet Map data set.

To solve this problem, we have routers mark packets using the last n bits *from the hash* of their IP addresses, rather than from their IP addresses alone. By modifying the $markingbits$ function to return the MD5 [27] cryptographic hash of the IP address, we achieve a nearly-uniform distribution of the last bits of the hash. In an actual router implementation of Pi, the router computes the MD5 hash only once and not on a per-packet basis.

4.3 Edge Marking in Pi

We now describe a mechanism to increase the entropy in an individual router's marking. Consider the fan-in topology shown in Figure 4. We compute the probability that the victim cannot distinguish the markings of a packet that traverses routers $R1$ and $R3$ from the markings of a packet that traverses routers $R2$ and $R3$. Let $M(Ri)$ be the n-bit marking that router Ri inserts. Assume that the n-bit marking is computed by the hash mechanism described in Section 4.2 above. Because router *R3*'s marking will be present in both paths, the probability that the markings for the two paths will be indistinguishable is equal to the probability that the markings of routers $R1$ and $R2$ are equal:

$$P[M(R1) = M(R2)] = \frac{1}{2^n}$$

If router $R3$ would adjust its marking, depending on whether the packet came from router $R1$ or $R2$, then the probability that we can distinguish the two paths increases. Suppose that router $R3$ marks the edge between the last-hop router and itself such that packets arriving from a router, RX, will be marked with $M(RX \rightarrow R3) = H(RX \,||\, R3)$, where the function H returns the n least significant bits of the MD5 hash and '$||$' represents concatenation. The probability that the two paths have the same marking now becomes:

$$\begin{aligned} P[\ \ & (M(Ri \rightarrow R1) = M(Rj \rightarrow R2)) \wedge \\ & (M(R1 \rightarrow R3) = M(R2 \rightarrow R3))\ \ ] = \tfrac{1}{2^n} \cdot \tfrac{1}{2^n} \\ & \qquad\qquad\qquad\qquad\qquad\qquad\quad = \tfrac{1}{2^{2n}} \end{aligned}$$

Edge marking thus decreases the probability that the two paths have the same marking by a factor of 2^n. We therefore adopt the edge marking scheme in Pi by changing the $markingbits$ function call in Figure 2 from $markingbits(Curr\_IP)$ to $markingbits(Curr\_IP, Prev\_IP)$ and pass the IP address of the last-hop router, $Prev\_IP$, as an additional argument to the $Pimark$ function.[2]

4.4 Suppressing Nearby Router Markings

The limited space in the IP Identification field causes routers close to the victim to overwrite the markings of routers farther away (assuming that the router path is sufficiently long that the TTL mark insertion pointer wraps around) from the victim. Unfortunately, the common markings of routers nearby to the victim may overwrite the distinguishing markings from routers farther away, which causes many initially distinct paths to end with the same Pi marking at the victim. We would like a mechanism that suppresses routers close to the victim from marking packets.

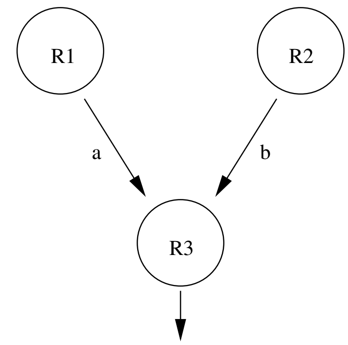


Figure 4. A Fan-In Topology.

A simple mechanism to achieve this would be to have a router *not* mark a packet if the destination IP address of that packet matches a route obtained through an Interior Gateway Protocol (IGP). The Internet is composed of many Autonomous Systems (AS) that run a variety of IGP routing protocols internally (such as OSPF or RIP), and then export address prefixes externally using the Border Gateway Protocol (BGP). The use of BGP has the effect of keeping routing tables small at lower tier ISP networks, which only need to know internal routes and a single route to all external addresses. This protocol shift is useful to us because it marks the boundary of the destination's AS. Thus, once the route to a destination is obtained through an IGP, all further routers in the path to the victim are within that AS and under the control of a single entity; which can presumably monitor local traffic in a more direct way than a generalized, Internet scale, packet marking scheme can. The important contribution of this improvement is that it extends the perimeter of our marking scheme from the victim to the AS boundary, allowing node markings from routers closer to the attacker to be preserved rather than overwritten by local AS routers for which traffic analysis mechanisms may already be in place.

However, there is a drawback to extending the non-marking router perimeter too far from the victim. To illustrate this, we take an attacker sending packets with a randomized IP Identification field. Normally, this is a weak attack, since the initial data in the IP Identification field is overwritten by router markings along the path to the victim. However, for u unmarked bits that reach the victim, the attacker can alternate between 2^u different packet markings. Thus, there is a tradeoff between extending and contracting the non-marking router perimeter, since this attack is only of concern for attackers who are closer than $\lfloor 16/n \rfloor$ *mark-*

[2] Note that the use of edge marking in Pi is different from its use by Savage et al. [28, 29]. In Pi, we use edge marking to increase the entropy of router markings, while Savage et al. use edge marking to enable path reconstruction.

ing routers to the victim. We note that it is a desireable property of Pi to force attackers to attack from areas nearby in the network topology, in order to be successful, because it is assumed that victims have more control or are able to better filter traffic from networks that are closer to them.

5 Filtering Schemes

This section describes how the victim can make use of the Pi marks to filter incoming packets during a DDoS attack. In Section 5.1 we present a basic, simple filter strategy. In Section 5.2, we discuss an attack that an intelligent adversary can execute on a victim using this filter, and present a countermeasure called *TTL Unwrapping* to defend against it. In Section 5.3 we present a more sophisticated filter based on the concept of thresholds. The design space of possible filter algorithms is quite large, and we discuss some advanced filter designs in Section 7.1.

5.1 The Basic Filter Scheme

The most basic filter a victim can apply to packets with Pi markings is to record the markings of identified attack packets and drop subsequent incoming packets matching any of those markings. Although this filter provides little flexibility to the victim, it has a very fast attack reaction time, since all the victim has to do is classify a single packet as an attack packet before being able to filter out all subsequent packets sent by that attacker. This filter also requires few memory resources, as it can be implemented in as little as 8Kbytes with a bit vector of length 2^{16} where the i'th bit of the vector is 0 if packets with a Pi mark of i are to be accepted and a 1 if packets with a Pi mark of i are to be dropped.

5.2 TTL Unwrapping

We must assume that DDoS attackers will do everything possible to increase the effectiveness of their attacks. Just as attackers use IP address spoofing to evade current packet filters, we assume that intelligent attackers will attempt to alternate between different packet markings at the victim so that their packets will not be easily identified. In Section 4.4 we examine the possibility of an attacker randomizing its IP Identification field to shift between different markings. However, this attack would only benefit attackers closer than $\lfloor 16/n \rfloor$ marking routers to the victim. A stronger attack, which can be effective regardless of the attacker's distance from the victim, is one where the attacker adjusts the initial TTL of its packets.

The attacker can modify the initial TTL of its packets to have the first hop router start marking in any one of the $\lfloor 16/n \rfloor$ sections of the IP Identification field. Because all routers along the path use the current TTL of the packet to determine the location in the IP Identification field to add their marking, the attacker can shift between $\lfloor 16/n \rfloor$ different markings just by changing its initial TTL. However, this TTL attack has only allowed the attacker to shift the marking bits in its packets to the left or right, not to change their individual values or relative ordering. We can use this to our advantage to devise a countermeasure.

When the victim server (or the first non-marking node in the case of marking suppression based on IGP routes) receives a packet, it can examine the TTL value and use it to find the oldest marking in the packet. This is the marking that would be overwritten if the victim were to mark the packet itself. The victim can use this value to unwrap the bits of the packet by rotating them so that the oldest marking is always in the most significant bit position. Thus, no matter what initial value the attacker chooses for its packets' TTL, the markings are always justified so that the oldest marking in the packet appears in a constant location. We call this mechanism *TTL Unwrapping* and assume that all victims implement it in addition to any of the other filters we discuss.

5.3 Threshold Filtering

There is another attack on our filtering strategy, which we call a *marking saturation* attack. In this attack, a large number of attackers spread throughout the Internet all send packets to a single victim in the hope of having the victim classify *every* marking as an attacker marking, and thus drop all incoming packets. This attack requires an attacker of immense means, since it requires *at least* 2^{16} zombie nodes, distributed in such a way that each attacker has a different Pi marking. However, despite the improbability of this attack, it does illustrate a weakness in our filtering strategy: a single attack packet with a particular marking can cause all other packets that share the marking to be dropped by the victim, regardless of the proportion of attack packets to legitimate user traffic. That is, the false negative rate (the rate at which user packets are dropped) of the basic filter is too high.

We introduce the notion of threshold filtering to allow the victim to lower the false negative rate at the expense of raising the false positive rate (the rate at which attack packets are accepted). The intuition behind threshold filters is that it may be in the victim's best interest to accept a small number of attack packets if that allows it to accept a large number of legitimate users' packets. The threshold filter is simply a value t_i (where $0 \leq i < 2^{16}$) chosen by the victim for each possible Pi mark value, i. Given the number of attack packets with Pi mark i, a_i and the number of user packets with Pi mark i, u_i, if the ratio of attack packets to total packets is such that:

$$\frac{a_i}{a_i + u_i} > t_i$$

then the victim drops all packets with Pi mark i. For example, a threshold value of $t_3 = 0.25$ would allow a victim to admit all packets with a Pi mark of 3, provided that attacker traffic comprises less than 25% of all traffic with a Pi mark of 3. We implemented a global version of the threshold filter, where all t_i equal a set value, T. Note that the basic filter is a special case of the global threshold filter with $T = 0$. We tested the filters' performance at T = 0%, 25%, 50% and 75%. We discuss these experiments and their results in Section 6.

6 Experimental Performance

In this section, we evaluate Pi's performance under DDoS attack. In order to evaluate Pi, we first describe our sample Internet data sets in the following section. We then explain the specific parameters that we choose for the design variables of our Pi scheme. We next present our DDoS attack model and the performance metrics that we measure. Finally, we present the results of our experiments and apply the experiments to incremental deployment scenarios.

6.1 Internet Data Sets

In our experiments, we use two Internet topologies: Burch and Cheswick's Internet Map [4, 14] and CAIDA's Skitter Map [6]. Both topologies were created by using a single host send `traceroutes` to hosts throughout the Internet and recording the paths as the IP addresses of the routers along each route. We filtered the data sets to remove all incomplete routes and duplicate routes (although multiple routes to the same end-host were not removed). We also removed all routes of path length shorter than seven hops.[3] In our experiments, we take the `traceroute` source of each map as the victim of our DDoS attack, and the end-hosts on the traceroute paths as our legitimate users and attackers.

6.2 Pi Parameter Selection

In this section, we explain our selection of the parameters n, the number of bits per router mark; and o, the number of hops away from the victim at which we suppress packet marking.

In choosing n, we consider only $n = 1$ and $n = 2$ bit schemes. There are two reasons for this choice. The first reason is that we would like our packet marking to carry information from a significant number of routers in the path. In choosing what is significant, we decided that at least a third of the routers in the path must contribute marks that reach the victim. Otherwise, we are in danger of having a marking scheme that provides very detailed edge markings, but only for four or less hops away from the victim. The last four hops are of limited value because they are likely to originate in a transit domain from which a large percentage of the victim's traffic arrives anyway (it would do little good for an e-commerce DDoS victim to drop all traffic from an Internet backbone, since the backbone is likely carrying most of its customer base). We decided that fewer than 5 hops of information (that is $n > 3$) would not provide sufficient detail to differentiate attack and legitimate users' packets. One reason to eliminate $n = 3$ is because only 15 bits would be used, leaving one bit unused. The loss of one bit may not seem significant, but it would limit our marking space to $2^{15} = 32768$ possible markings. This limitation would make the effects of *marking saturation* more severe than they otherwise would be, so we eliminated $n = 3$ as an option as well. The number of bits per router marking n, must be a globally imposed constant in a deployed Pi system. We chose $n = 1$ and $n = 2$ based on the Skitter [6] and Internet Map [14, 4] data sets. It is possible that the real Internet topology may be substantially different from this sample data, in which case our choices will have to be revisited.

Unlike the choice of n, the value for o — the number of hops away from the victim at which routers stop marking — is not a globally imposed constant. Rather, each organization can decide on the best value and configure the routers within its control accordingly. The benefit of choosing a large o value is that earlier markings (closer to the attacker or user) will not be overwritten by the routers close to the victim, which presumably handle the majority of the victim's traffic anyway. However, choosing a large o has a drawback as well. By pushing the perimeter of non-marking routers farther from the victim, the number of routers that mark the packet is reduced accordingly. Thus, it is more likely that randomized attacker initialized markings will remain in the packet, thus allowing an attacker to alternate between markings even when on the same path. These two contrasting characteristics cause us to pick different o values depending on the n value that is chosen. For $n = 1$ we need 16 routers to completely overwrite any attacker initialized data. Thus, we would like a small o value, so that as many routers as possible will mark the packet. We therefore choose $o = 0$ (where all routers, except the victim itself, mark packets) for our tests with $n = 1$. For $n = 2$, however, we would like as large an o value as possible, since only 8 markings fit into a single packet and that is well short of the average path length of 15 for our data sets. Unfortunately, it is difficult to determine exactly what a reasonable number of hops away from a victim are still under that victim's administration. Therefore, we have chosen a value of $o = 3$, based on limited data from several `traceroutes` that we have performed to large web servers (like Amazon.com),

[3] Although this step will slightly bias the experimental results in our favor, it was initially necessary to simplify our simulation code. Furthermore, the number of paths removed is insignificant compared to the total number of paths in the data sets.

for testing under an $n = 2$ bit scheme.

6.3 DDoS Attack Model

In order for a DDoS victim to protect itself against attack packets, it must have a way to differentiate them from normal user packets. In Section 3 we state that such a method was required to identify the packet markings that the victim will add to its attack markings list. Once those markings are identified, it is a simple matter to drop packets with the same marking by comparing incoming packet markings against the markings in the attack list. It is outside the scope of this paper to define an algorithm for attack packet identification. Jung, Krishnamurthy and Rabinovich further discuss this problem in [19]. Assuming such an algorithm, we model our DDoS attack in two phases. In the first phase, the *learning phase*, all packets are assumed to be analyzed by the victim, using the packet identification function that determines whether the packet is an attack packet or a legitimate user's packet. In other words, the victim is temporarily given the power to differentiate between legitimate users' packets and attackers' packets. The victim is thus able to generate an attack markings list. In the second phase, the *attack phase*, the victim is presumably no longer able to apply its packet identification function and is forced to use the Pi filter based on the information it has gathered in the learning phase.

6.4 Experiment Design and Performance Metrics

For our experiments, we choose 5000 paths at random from one of our Internet data sets to act as legitimate users. We choose our attackers in the same way, but with the constraint that attackers and users are disjoint. Each end-host at a path, whether user or attacker, sends three packets to the victim server in phase one of the attack, and three packets in phase two of the attack. We choose a three packet learning phase to illustrate how quickly Pi filters can react to DDoS attacks. A longer learning phase (which would almost certainly be the case in a real deployment scenario) would only improve performance further, because the victim would have more packet markings on which to base its filtering decisions. As our performance metric, we calculate the ratio of the number of attack packets accepted by the victim to the total number of attack packets sent (the attacker packet acceptance ratio) as well as the ratio of the number of user packets accepted by the victim to the total number of user packets sent.[4] In some of our results we show the acceptance ratio *gap*, which is simply the attacker packet acceptance subtracted from the user packet acceptance rate. If the victim server where to apply a completely random filter, then the user and attacker packet acceptance ratios would be exactly equal, so the acceptance ratio gap provides a metric that shows how much better the Pi filter is performing compared to no filter at all. Our results are presented in Figures 5 through 8 and discussed in the next two sections.

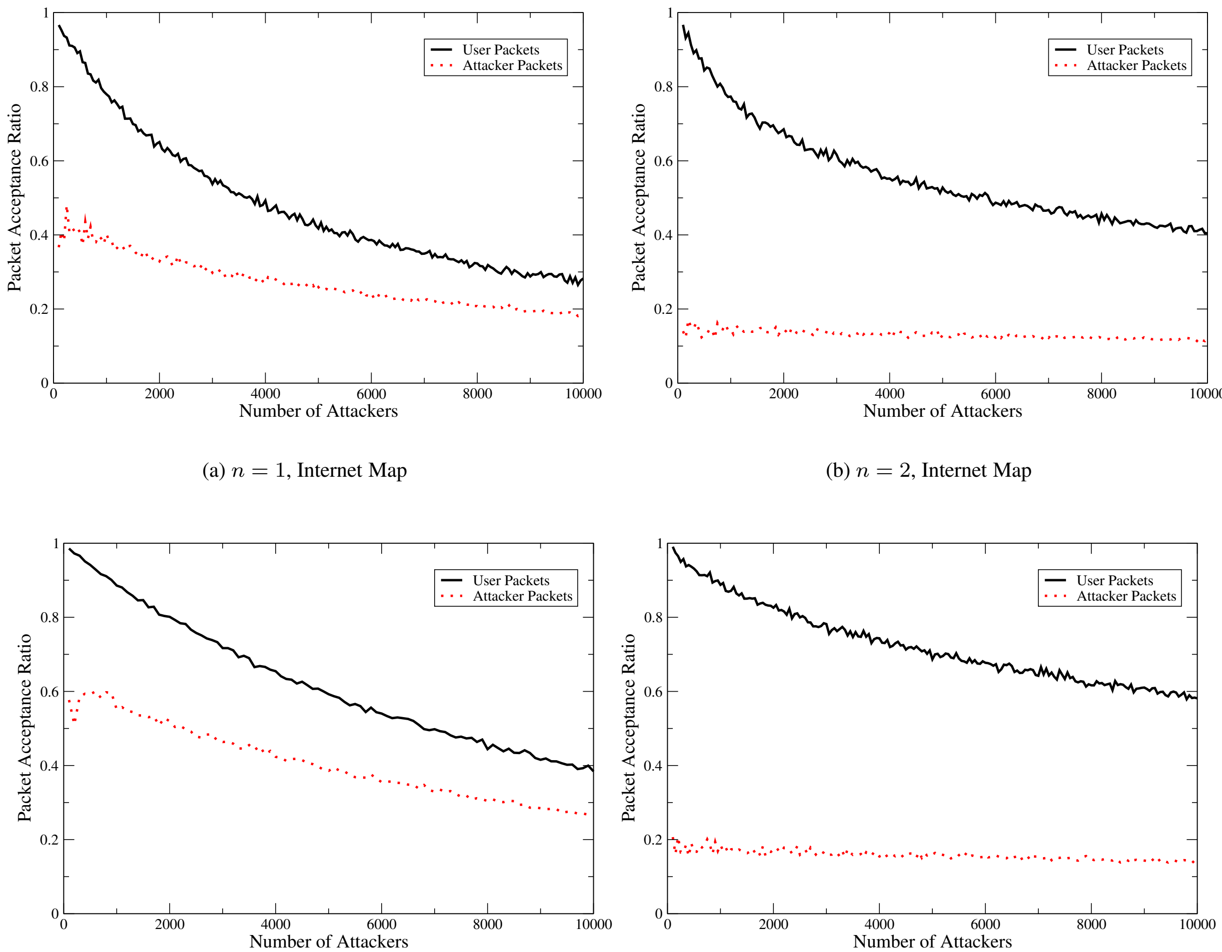
6.5 Results

In Figure 5 we see the $n = 1$ bit and $n = 2$ bit schemes with the basic Pi filter (equivalent to a threshold of T = 0). These curves represent the strictest possible filtering in the Pi scheme: a single attack packet with a particular marking received during the learning phase of the DDoS attack causes all packets with that marking to be dropped during the attack phase. The attack packet acceptance ratio is due to attackers located near enough to the victim that the random data that they initialize into the IP Identification field of their packets is not completely overwritten, allowing them to alternate to markings that were not recorded by the victim in the learning phase of the attack. Because the $n = 1$ bit scheme requires twice the number of marking routers as the $n = 2$ bit scheme to overwrite such random data, its attacker acceptance ratio is larger.

The downward slope exhibited for the user acceptance ratio, in both schemes, is due to the increasing number of attacker markings that collide with user markings, causing them to be dropped. This is an example of the *marking saturation* effect which we discuss in Section 5.3. Surprisingly, marking saturation also affects attackers as well as legitimate users, as exhibited by the downward slope of the attacker acceptance ratios in Figures 5a and 5c. With a larger number of attackers, attack packets begin interfering with each other, in the sense that an attacker a may shift between four markings, two of which another attacker, b, is also shifting between. Because both a and b send packets in the learning phase, it is more likely that the overlapping markings will be received by the victim and added to the attacker markings list than it would be if only one of the attackers is present. The downward slope is minimized for the $n = 2$ bit scheme in Figures 5b and 5d because there are fewer attackers that are close enough to the victim to shift between markings.

In Figure 6 we show the effect of increasing the threshold value to combat the marking saturation effect. In this experiment, we set the threshold value to 50%, where more than half of the packets arriving with a particular marking must be attack packets before the victim begins dropping all packets with that marking. Of course, increasing the threshold value increases the overall number of packets accepted,

[4] The packet numbers used in our metrics are taken only from phase two of the attack — after the attack packets have been identified. This is a reasonable measurement of our scheme's performance because no DDoS protection mechanism that we are aware of can stop attack packets before they are first classified as such (Ingress filtering, where deployed, can stop attack packets with spoofed source IP addresses, but still forces victims to identify malicious flows from attackers using legitimate source IP addresses).



(a) $n = 1$, Internet Map

(b) $n = 2$, Internet Map

(c) $n = 1$, Skitter Map

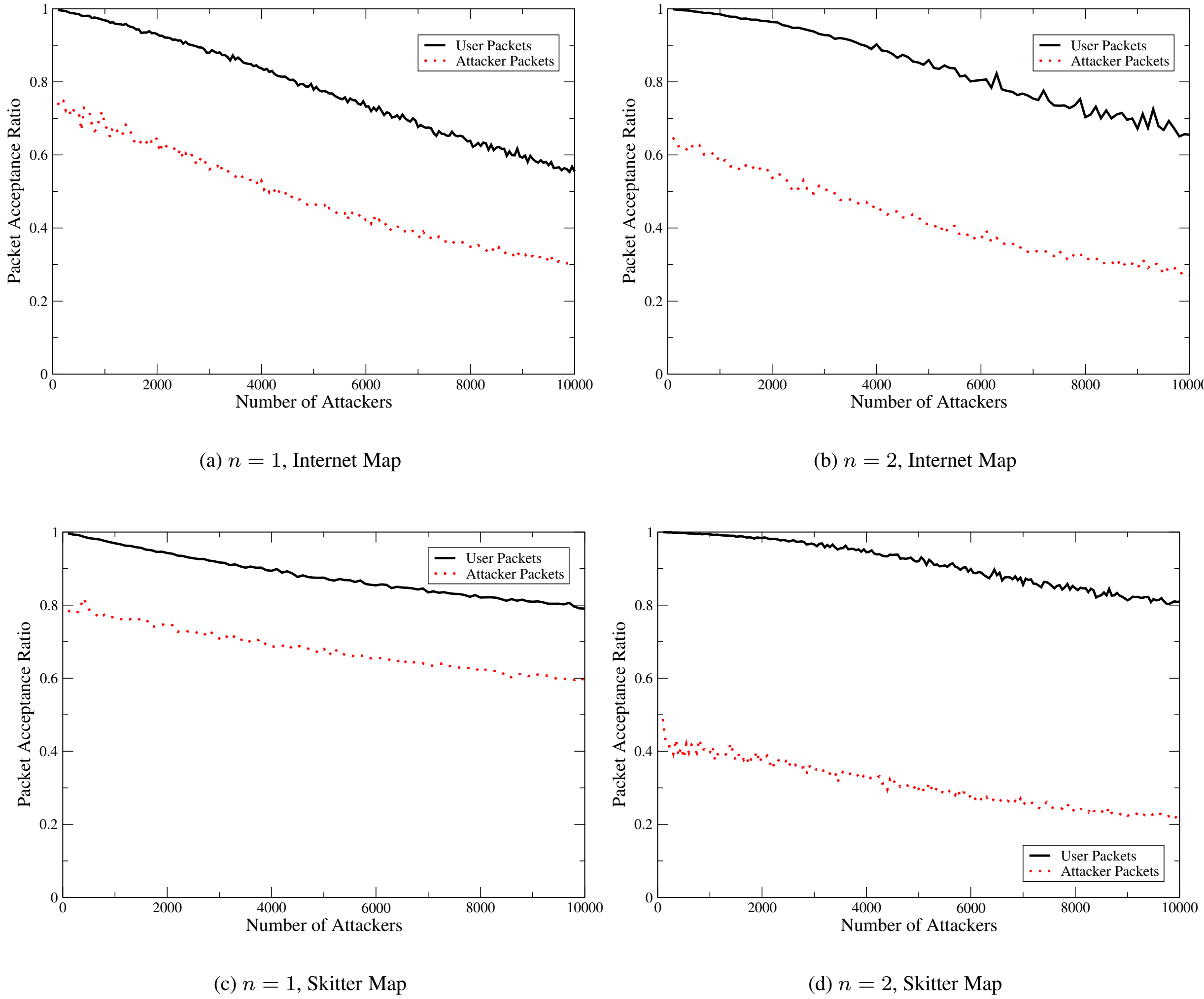
(d) $n = 2$, Skitter Map

Figure 5. Pi Filtering with a 0% Threshold

which is reflected in the higher acceptance ratios for both the users and attackers.

From our results comparing the 0% and 50% threshold values, we can confirm the intuitive result that raising the threshold value can minimize the marking saturation effect. With the 50% threshold, marking saturation affects attackers and users equally because simply receiving an attack packet with a particular marking at the victim no longer results in dropping all the users' packets with that marking. This phenomenon is shown in Figure 6 as the equal downward slope exhibited by both the attacker and user packet acceptance ratios. What this suggests is that victims may want to modify their threshold filter values according to the severity of an attack. In Figure 7 we plot the acceptance ratio gap for four different threshold values. This figure shows which thresholds should be used according to the severity of an attack. As the number of attackers increases, higher threshold values perform better than lower threshold values who's user acceptance ratios plummet because many markings are flagged as attack packets.

Overall, these results are promising, particularly for the $n = 2$ bit scheme. Pi filtering provides significant differentiation between user and attack packets after only a three packet learning phase, even when thousands of attack paths are used. Pi filtering with thresholds provides an adjustable mechanism to defend against attacks of varying severity. Finally, the behavior of the Pi scheme is consistent across both the Skitter and Internet Map datasets, which shows that Pi's performance is not limited to a single Internet topology.



(a) $n = 1$, Internet Map

(b) $n = 2$, Internet Map

(c) $n = 1$, Skitter Map

(d) $n = 2$, Skitter Map

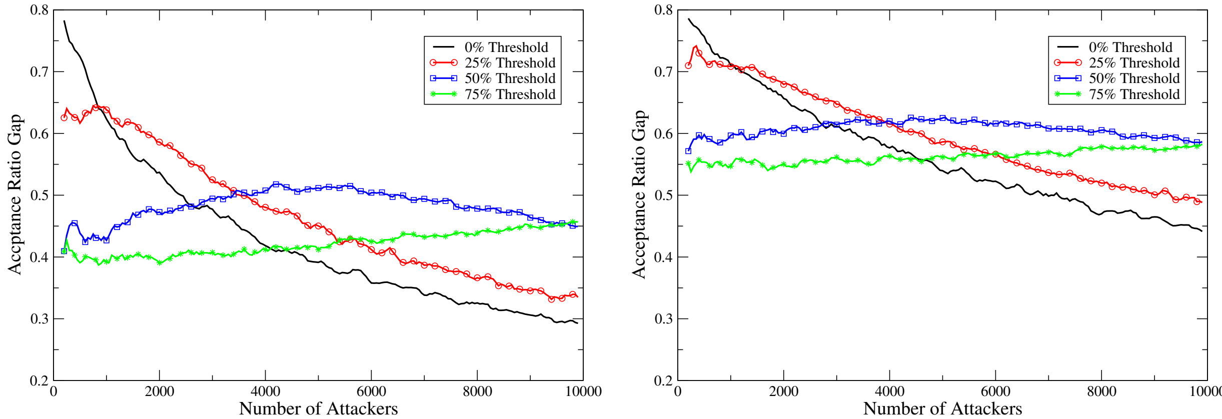
Figure 6. Pi Filtering with a 50% Threshold

6.6 Legacy Routers

Any proposed packet marking scheme must be robust to the presence of legacy routers. In Pi marking, legacy routers may cause unmarked bits to appear in the IP Identification field of the packet. The reason for this is that a legacy router will decrement the TTL of a packet, thus shifting the marking index by one, but will not mark anything into the field. Thus, an unmarked n number of bits is left in the packet. These unmarked bits may be harmlessly overwritten by Pi-enabled routers farther down the packet's path towards the victim, however, even in the $n = 2$ scheme, there are rarely enough marking routers in the packet's path for this to occur. More often, unmarked sections of the field make it to the victim, allowing the attacker to shift between 2^{nl} markings, where l is the number of unmarked sections arriving at the victim.

In Figure 8 we show the affect of increasing the percentage of legacy routers in our sample topologies. We run the same experiment as in the previous section, with the $n = 2$ bit scheme and a 50% threshold value, only this time we assign a probability p to each router in the topology that it will function as a legacy router. If a router is chosen as a legacy router, it acts as one for the entire simulation. We note that this uniform distribution of legacy routers is unlikely to represent the true properties of incremental deployment, since new routers implementing our scheme will likely be deployed in clusters, and initially only in the core routers of the Internet. However, a uniform distribution is actually more pessimistic in our scheme, since a continuous path of non-legacy routers is more likely to overwrite attacker generated random data then a scattered group of them, which may simply overwrite each other.

The results in Figure 8 show that the acceptance gap of the Pi filter is inversely proportional to the percentage of legacy routers in the topology. However, it is clear from



(a) $n = 2$, Internet Map

(b) $n = 2$, Skitter Map

Figure 7. Pi Filter Performance vs. Threshold Value

the graph that the Pi filter continues to provide some level of differentiation between user and attacker packets, even when only 50% of the routers in the sample topology actually participate in the marking scheme.

7 Discussion

In this section, we discuss more advanced filters, and how to use Pi to support other anti-DDoS mechanisms. In our technical report, we discuss more advanced marking and filtering techniques, alternative performance metrics, issues related to fragmentation, among others [26].

7.1 Advanced Filters

It seems apparent from our experiments that the $n = 2$ bit scheme is superior to the $n = 1$ bit scheme. Unfortunately, this may be a consequence of the relatively simplistic filtering process that we implement. The filters that we implement are largely static in that they select parameters that remain constant throughout the length of the simulation. A separate, potentially more powerful, class of filters are dynamic filters. An example of such a filter would be a longest prefix matching filter which could build a table of Pi marking prefixes based on incoming packets' markings. This filter would primarily benefit the $n = 1$ scheme, because a victim equipped with this filter might only use a certain number of marking bits available to it, rather than all 16 bits, which may contain some attacker initialized bits.

The Pi mechanism can also be used to detect spoofed IP addresses, with an appropriate filter. The victim need only build a table correlating the Pi mark of a packet to its source IP address, during non-attack time. When under attack, the victim can check to see if the source IP addresses of incoming packets match against the IP addresses of their Pi marks from the table.

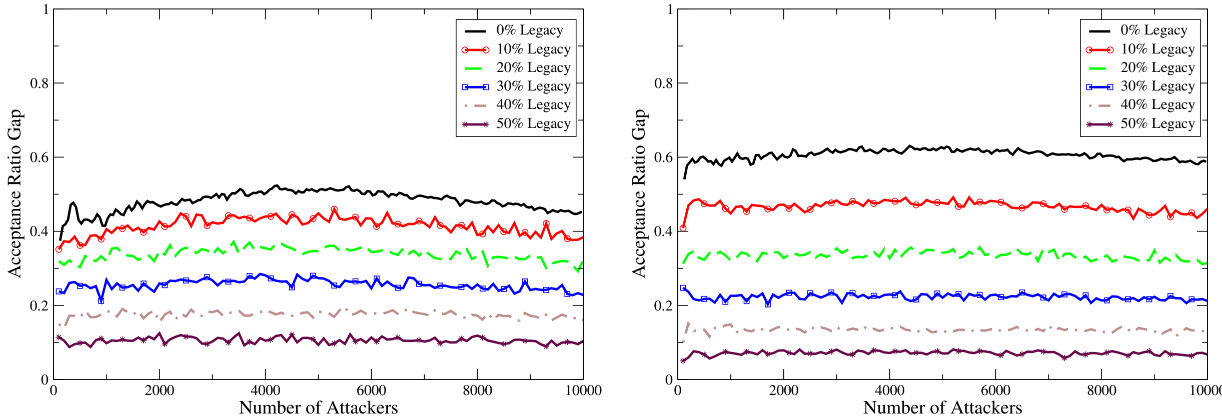
There are many potential uses for a Pi filter that detects spoofed IP addresses. In a particular type of DDoS attack, known as a *reflector attack* [25], attackers send request packets to various services whose responses are of far larger size than the requests themselves (e.g., DNS). The attackers spoof the source IP address of the requests as the IP address of the intended victim of the attack. Thus, the service's reply packets will be sent to the victim, with the additional benefit of the traffic amplification of the responses. Thus, the machine supplying the service is used as a *reflector*, focusing and amplifying the traffic to the victim. A Pi filter capable of detecting spoofed IP addresses running on on the reflector's server would immediately detect the spoofed source IP addresses of the requests and refrain from sending a response, thus halting the attack.

The IP spoofing detection filter can also be used for a limited form of traditional IP Traceback — given a Pi mark, the victim can check the list of IP addresses from the table that match that mark and simply perform `traceroutes` to those IP address.

Clearly, the design space for possible Pi filters is quite large and remains an open research topic. However, we note that the Pi filter mechanisms are end-host specific and can be changed even after the Pi marking scheme is deployed at routers.

7.2 Filtering in the Network

The Pi marking scheme can also support other anti-DDoS systems. For example, the Pushback system [16, 21]



(a) $n = 2$, 50% Threshold, Internet Map

(b) $n = 2$, 50% Threshold, Skitter Map

Figure 8. Legacy Router Performance

uses downstream routers that identify *aggregates* (packets from one or more flows that have certain characteristics, such as source or destination addresses) and send rate-limit requests to upstream routers, along with an aggregate identifier. The problem with this technique is that DDoS packets may share little identifying traits, beyond the destination IP address. However, using the Pi marking, each router can identify common markings (after applying TTL Unwrapping) and use these to better identify particular aggregates.

The Pi marking can also be used to move Pushback filters closer to the attacker, as the marking is an identifier of the path towards the attacker. However, the Pushback router needs to consider that the Pi markings are not unique, as multiple paths may exhibit the same marking.

8 Related Work

We first discuss general papers on network DoS. Moore, Voelker, and Savage use *backscatter* packets (the unsolicited responses that a DoS victim sends to the spoofed IP address that it receives in the attack packet) to gauge the level of Internet DoS activity [24]. Jung, Krishnamurthy, and Rabinovich attempt to answer the question of how a site can differentiate between a DoS attack and a simple high load condition by analyzing client request rates and file access patterns [19].

Many approaches for securing against DoS and DDoS attacks are present in the literature. Early methods focused on detecting the ingress and egress points of DoS traffic within a single network administration. Ferguson and Senie propose to deploy network ingress filtering to limit spoofing of the source IP address [13]. A more recent and functional approach to ingress filtering is proposed by Li et al. in [20]. Their protocol, called SAVE, has routers construct tables of valid source addresses per incoming interface, in much the same way that they construct routing tables of destination addresses per interface. A packet whose source address is out of the proper range is easily identified and dropped. Stone proposes the CenterTrack mechanism, which uses routers capable of input debugging (the ability to identify through which router interface a particular packet was received) that would be virtually connected through IP tunnels to all border routers on a network [35]. When a node in the network comes under attack, the overlay network is activated, and all border routers channel traffic through the overlay routers. These routers would use input debugging to tell from which border router, and hence which neighboring network, the DoS traffic is coming from.

Burch and Cheswick present another scheme for path tracing [5]. This unique scheme uses a limited form of DoS attack to attack the exact path which the DoS traffic is traversing. By selectively exhausting select network resources and monitoring the perturbations in the DoS attack traffic, it is possible to detect the links that a DoS attack is traversing. Unfortunately, this method does not scale well for the multiple attackers in a DDoS attack, nor does it solve the problem of administrative coordination between ISPs.

None of the methods described previously relies on the IP protocol to assist in protecting against DoS attacks. A new class of protections seeks to modify parts of the IP protocol itself to assist in finding the path of DoS and DDoS traffic. Early works in this category suggest adding a new type of ICMP message: traceback messages [3, 17]. For each packet received, routers would, with a small probability, generate an ICMP message to the destination address of the packet containing the IP address of the router. The problem with this initial scheme is that there is a tension between providing fast (in the number of packets received at the victim) path identification, and low network overhead generated by added messages. Mankin et al. present an improvement to this scheme, which puts some state on the routers to generate better traceback messages [22]. Better traceback messages are defined as the ones originating from routers that are far away from, and that have not been seen previously by, the victim. Although this improvement reduces the overhead of ICMP traceback significantly, it relies on either a shared key distribution mechanism to prevent attacker forged traceback messages (which is a very difficult problem on an Internet scale); or on asymmetric cryptography, which could potentially be exploited by attackers in a DoS attack by exhausting server resources with failed, yet time-consuming signature verifications.

Several researchers propose to embed traceback information within the IP packet. Savage et al. first proposed this approach [28, 29]. They use the 16 bit IP Identification field to hold traceback information, probabilistically generated by routers along the packet's path. A particular router marks a fragment of its IP address and sets a bit to signal the next router to do the same, thus marking a fragment of the edge between those two routers. Fragments may be overwritten by other routers farther down the path toward the victim. Fragments are reassembled at the victim to reconstruct all the IP addresses of the upstream routers towards the attacker. This method works well for DoS attacks with few attackers; however in DDoS attacks, fragment reconstruction at the victim becomes computationally expensive. Song and Perrig show how to overcome this hurdle by having a map of the upstream routers present at each victim [33]. However, the victim must still receive on the order of one thousand packets to identify the attack path. Dean, Franklin, and Stubblefield propose using algebraic codes to encode the upstream router path for IP traceback [8, 9]. Nodes mark packets with evaluations of the sample points of a polynomial over a finite field. The coefficients of the polynomial are the IP addresses of the routers in the attack path.

Adler presents an ingenious scheme for sampling the frequency of an x-bit number to determine the paths that packets are taking [1]. Routers assign themselves a 0 or 1 bit based on whether they are at the left or right branch of the next upstream router (although a binary tree topology is assumed, this assumption can be relaxed). Based on the incoming bit marking of a packet, and the self-assigned bit of the router, each router has a certain probability of marking a 0 or 1 bit in the packet. In the case of more than one bit, the path is split into x smaller paths, each one of which executes the one bit protocol in a separate bit position in the packet. Like the other probabilistic methods though, this scheme does not scale well to multiple paths of attack in that it requires an exponentially increasing number of packets to accurately judge the attacking paths.

Sung and Xu present a similar method to Pi marking that allows the victim to participate in packet filtering [36]. Their approach utilizes existing IP Traceback mechanisms, but introduces the concept of preferential packet filtering. In their scheme, a small subset of packets carry IP Traceback information and the majority of packets are probabilistically marked with the hash of network edges. While the victim is reconstructing the attack graph using the IP Traceback packets, it can apply packet filtering to the edge markings of packets based on whether or not they are likely to appear in the attack graph.

To surmount the problem of large numbers of packets necessary at the victim to traceback multiple attack paths, Snoeren et al. propose a solution using router state to track the paths of a single packet [31, 32]. Upon receipt of a packet, each router hashes specific, invariant fields of the packet and stores the hash in a table. When traceback is needed, the victim presents its upstream router with the hash of the packet to be traced. The routers at each hop away from the victim then recursively query the routers at the next hop away for the presence of the hash of the packet in their hash tables. Besides the ability to traceback single packets, this method also offers the advantage of storing saturated hash tables for traceback after an attack has taken place. Duffield and Grossglauser propose using packet hashes of a subset of all network traffic to assist in traffic measurement in a network [11]. In this method, called *trajectory sampling*, packets are hashed, deterministically, and a subset of them are sampled at every node in the network that they traverse. These samples are all sent to a centralized measurement system, which can reconstruct the packets' paths through the network.

Ioannidis and Bellovin, and Mahajan et al. propose Pushback, a packet filtering infrastructure leveraging router support to filter out DDoS streams [16, 21]. We discuss their work and its potential synergy with Pi in Section 7.2.

9 Conclusion

In this paper, we have presented Pi, a novel approach to defend against DDoS attacks. Our proposal draws from elements of IP Traceback methods but is not concerned with reconstructing a path from a victim to an attacker, rather, it is concerned with marking paths with unique markings. This gives the victim of a DDoS attack the ability to filter, on a per-packet basis, any incoming packets that match known attacker marks.

We have shown how to increase the entropy of the Pi marking by utilizing several improvements, specifically: IP address hashing to obtain a uniform distribution of packet marks per node; node omission based on the presence of intra-AS routes to increase the number of distant routers whose markings arrive at the victim; and edge marking to lower the probability of collisions of different paths. We have also secured our marking method against attacker modified TTL values by utilizing TTL Unwrapping, which uses the TTL value at the victim to rotate the bits of a packet's marking to a standard position, irrespective of the initial TTL.

We establish a model for DDoS attacks that consists of two phases: the learning phase and the attack phase. We run experiments that simulate a DDoS attack on a server with a constant 5000 user load and a variable number of attackers, from 100–10000. We show the performance of two marking schemes, $n = 1$ bit and $n = 2$ bits, using a threshold filtering mechanism. We show that both schemes provide good protection against DDoS, and degrade gracefully under added attacker load.

Finally, we demonstrate that the Pi marking scheme has strong incremental deployment properties, such that a victim is still able to filter incoming packets even when 50% of routers in our topology do not participate in the marking. We believe that Pi marking is the most general, flexible, and powerful of the packet marking schemes to date, and shows significant potential in reducing or eliminating the DDoS threat.

10 Acknowledgments

We would like to thank Mike Reiter for their discussions and feedback on this project. We would also like to thank the anonymous reviewers for their insightful comments and suggestions.

References

[1] M. Adler. Tradeoffs in probabilistic packet marking for IP traceback. In *Proceedings of 34th ACM Symposium on Theory of Computing (STOC)*, 2002.

[2] Associated Press. Internet attack was much worse than anticipated. *foxnews.com*, Jan. 2003. `http://www.foxnews.com/story/0,2933,76804,00.html`.

[3] S. Bellovin, M. Leech, and T. Taylor. The ICMP traceback message. Internet-Draft, draft-ietf-itrace-01.txt, Oct. 2001. Work in progress, available at `ftp://ftp.ietf.org/internet-drafts/draft-ietf-itrace-01.txt`.

[4] H. Burch and B. Cheswick. Internet watch: Mapping the Internet. *Computer*, 32(4):97–98, Apr. 1999.

[5] H. Burch and B. Cheswick. Tracing anonymous packets to their approximate source. Unpublished paper, December 1999.

[6] Caida. Skitter. `http://www.caida.org/tools/measurement/skitter/`, 2000.

[7] Computer Emergency Response Team (CERT). TCP SYN flooding and IP spoofing attacks. Technical Report CA-96:21, Carnegie Mellon University Pittsburgh, PA, Sept. 1996.

[8] D. Dean, M. Franklin, and A. Stubblefield. An algebraic approach to IP traceback. In *Network and Distributed System Security Symposium (NDSS '01)*, February 2001.

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

[10] D. Dean and A. Stubblefield. Using client puzzles to protect TLS. In *Proceedings of the 10th USENIX Security Symposium*, Aug. 2001.

[11] N. G. Duffield and M. Grossglauser. Trajectory sampling for direct traffic observation. In *Proceedings of the 2000 ACM SIGCOMM Conference*, Aug. 2000.

[12] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In E. F. Brickell, editor, *Advances in Cryptology – Crypto '92*, pages 139–147, 1992.

[13] P. Ferguson and D. Senie. Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing. RFC 2267, January 1998.

[14] Internet mapping. `http://research.lumeta.com/ches/map/`, 2002.

[15] Internet Software Consortium. Internet domain survey host count. `http://www.isc.org/ds/hosts.html`, July 2002.

[16] J. Ioannidis and S. M. Bellovin. Implementing Pushback: Router-based defense against DDoS attacks. In *Proceedings of the Symposium on Network and Distributed Systems Security (NDSS 2002)*, Feb. 2002.

[17] ICMP traceback (itrace). IETF working group, `http://www.ietf.org/html.charters/itrace-charter.html`.

[18] A. Juels and J. Brainard. Client puzzles: A cryptographic defense against connection depletion attacks. In *Proceedings of the 1999 Network and Distributed System Security Symposium (NDSS '99)*, pages 151–165, 1999.

[19] J. Jung, B. Krishnamurthy, and M. Rabinovich. Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites. In *The Eleventh International World Wide Web Conference (WWW 11)*, May 2002.

[20] J. Li, J. Mirkovic, M. Wang, P. Reiher, and L. Zhang. SAVE: Source address valididty enforcement protocol. In *Proceedings of IEEE INFOCOMM 2001*, Apr. 2001.

[21] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling high bandwidth aggregates in the network. *CCR*, 32(3):62–73, July 2002.

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

[23] D. McGuire and B. Krebs. Attack on internet called largest ever. *washingtonpost.com*, Oct. 2002. `http://www.washingtonpost.com/wp-dyn/articles/A828-2002Oct22.html`.

[24] D. Moore, G. Voelker, and S. Savage. Inferring internet denial of service activity. In *Proceedings of the 10th USENIX Security Symposium*, Aug. 2001.

[25] V. Paxson. An analysis of using reflectors for distributed denial-of-service attacks. *Computer Communication Review*, 31(3), 2001.

[26] A. Perrig, D. Song, and A. Yaar. Pi: A new defense mechanism against IP spoofing and DDoS attacks. Technical Report CMU-CS-02-207, Carnegie Mellon University, School of Computer Science, Dec. 2002.

[27] R. L. Rivest. The MD5 message digest algorithm. RFC 1321, Internet Activities Board, Internet Privacy Task Force, Apr. 1992.

[28] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for IP traceback. In *Proceedings of the 2000 ACM SIGCOMM Conference*, August 2000.

[29] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Network support for IP traceback. *ACM/IEEE Transactions on Networking*, 9(3), June 2001.

[30] C. L. Schuba, I. V. Krsul, M. G. Kuhn, E. H. Spafford, A. Sundaram, and D. Zamboni. Analysis of a denial of service attack on TCP. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, May 1997.

[31] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer. Hash-based IP traceback. In *Proceedings of the ACM SIGCOMM 2001 Conference*, pages 3–14, Aug. 2001.

[32] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, B. Schwartz, S. T. Kent, and W. T. Strayer. Single-packet IP traceback. *IEEE/ACM Transactions on Networking (ToN)*, 10(6), Dec. 2002.

[33] D. X. Song and A. Perrig. Advanced and authenticated marking schemes for IP traceback. In *Proceedings of IEEE INFOCOMM 2001*, April 2001.

[34] I. Stoica and H. Zhang. Providing guaranteed services without per flow management. In *Proceedings of the 1999 ACM SIGCOMM Conference*, Apr. 1999.

[35] R. Stone. CenterTrack: An IP overlay network for tracking DoS floods. In *Proceedings of the 9th USENIX Security Symposium*, Denver, Colorado, Aug. 2000. USENIX.

[36] M. Sung and J. Xu. IP traceback-based intelligent packet filtering: A novel technique for defending against internet DDoS attacks. In *Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP'02)*, Nov. 2002.