StackPi: New Packet Marking and Filtering Mechanisms for DDoS and IP Spoofing Defense

Abraham Yaar Adrian Perrig Dawn Song
Carnegie Mellon University
{ayaar, perrig, dawnsong}@cmu.edu

***Abstract*— Today's Internet hosts are threatened by large-scale Distributed Denial-of-Service (DDoS) attacks. The Path Identification (Pi) DDoS defense scheme has recently been proposed as a deterministic packet marking scheme that allows a DDoS victim to filter out attack packets on a per packet basis with high accuracy after only a few attack packets are received [40].**

In this article, we propose the StackPi marking, a new packet marking scheme based on Pi, and new filtering mechanisms. The StackPi marking scheme consists of two new marking methods that substantially improve Pi's incremental deployment performance: Stack-based marking and Write-ahead marking. Our scheme almost completely eliminates the effect of a few legacy routers on a path, and performs 2–4 times better than the original Pi scheme in a sparse deployment of Pi-enabled routers. For the filtering mechanism, we derive an optimal threshold strategy for filtering with the Pi marking. We also develop a new filter, the *PiIP filter*, which can be used to detect IP spoofing attacks with just a single attack packet.

Finally, we discuss in detail StackPi's compatibility with IP Fragmentation, applicability in an IPv6 environment, and several other important issues relating to potential deployment of StackPi.

***Index Terms*— Security, system design, distributed denial of service defense, DDoS.**

I. Introduction

I-A. IP Spoofing and DDoS Attacks

Internet security is of critical importance to our society, as the government and economy increasingly rely on the Internet to conduct their business, and people use the Internet as a convenient vehicle for simplifying a wide range of tasks, from banking to shopping. Unfortunately, the current Internet infrastructure is vulnerable to a Distributed Denial of Service (DDoS) attack. Because DDoS attacks typically rely on compromising a large number of hosts to generate traffic to a single destination, the severity of DDoS attacks will likely increase as greater numbers of poorly secured hosts are connected to high-bandwidth Internet connections. For example, an attacker who could compromise the popular SETI@Home [32] distributed computation software, or any popular P2P client, would be able to harness several hundreds of thousands of hosts to generate traffic for an attack.

The weakness of the current Internet infrastructure that facilitates DDoS attacks is the inability for a packet recipient to authenticate that packet's claimed source IP address. In other words, an attacker can intentionally modify, or *spoof*, the source address of the packets it sends from a compromised host. Two examples of DDoS attacks that rely on IP address spoofing are:

- TCP SYN Flooding: In this attack, an attacker sends TCP SYN packets as if to initiate a TCP connection with its victim. These SYN packets contain spoofed source IP addresses, which cause the victim to waste resources that are allocated to half-open TCP connections which will never be completed by the attacker [9].
- Reflector Attack: In this attack described by Paxson [30], the attacker attempts to overwhelm the victim with traffic, by using intermediate servers to amplify the attacker's bandwidth and/or hide the attacker's origin. The attacker simply sends requests to the intermediate server with a spoofed source IP address matching the victim's IP address. The intermediate server only sees that a number of requests are supposedly coming from the victim, and so sends its responses to the victim. When properly coordinated, a group of attackers can cause a flood of packets to hit the victim, without sending any packets directly to the victim itself. To amplify the traffic, the attacker selects intermediate servers whose responses to the spoofed requests are larger than the requests themselves. For example, in DNS server based reflector attacks, attackers send short DNS lookup requests (50 bytes each), whose replies can be over a thousand bytes long, thus giving the attacker a 20-fold traffic amplification. Other popular reflectors are Internet game servers, where attackers can use similar methods to gain two orders of magnitude of traffic amplification [21].

These types of DDoS attacks, which use large amounts of traffic to disable a victim server, are the focus of this article. However, source IP address spoofing is also used in many other attacks. An attacker who wants to evade source IP address based packet filtering will use source IP spoofing. Finally, some DDoS attacks do not rely on source IP address spoofing, because the attacker simply does not care whether or not the machine that it has compromised is implicated in the attack, so long as the attacker itself remains unknown. However, as source IP address filtering mechanisms become widely deployed (e.g., the Pushback framework [17], [25]), it is likely that attackers will have to resort to source IP address spoofing to increase the effectiveness of their attacks.

I-B. Desired Properties of Defense Mechanisms against DDoS Attacks

Because the current Internet infrastructure has few capabilities to defend against DDoS attacks, we need to design a new network level defense mechanism against these attacks. In

particular, a good solution to defend against these attacks should satisfy the following properties:

- Fast response: The solution should be able to rapidly respond to and defend against attacks. Every second of Internet service disruption causes economic damage. We would like to immediately enable blocking of attack traffic.
- Scalable: Some attacks, such as TCP SYN flooding, involve a relatively small number of packets. However, many DDoS attacks are large scale and involve thousands of distributed attackers. A good defense mechanism must be effective against low packet count attacks, but also scale up to handle large-scale attacks.
- Victim filtering: Some DDoS defense schemes in the literature assume that once the attack path is revealed, upstream routers will install filters in the network to drop attack traffic. This is a weak assumption because such a procedure may be slow, since the upstream ISPs have no incentive to offer this service to non-customer networks and hosts. A defense mechanism should enable sites to perform local filtering, which is especially effective if the attack does not cause network congestion.
- Efficient: The solution should have very low processing and state overhead for routers and, to a lesser degree, victim servers.
- Support incremental deployment: The solution is only useful and practical if it provides a benefit when only a subset of routers in the network implement it. As an increasing number of routers deploy the scheme, there should be a corresponding performance increase.

Many of the current DDoS defense schemes address only a subset of these properties. We review these schemes in detail in Section VII. Our Pi scheme satisfies all of the above properties.

I-C. Our Contributions

This article makes the following contributions. We propose the StackPi marking, a new packet marking scheme based on Pi, and new filtering mechanisms. The StackPi marking scheme consists of two mew marking methods that substantially improve Pi's incremental deployment performance: Stack-based marking and Write-ahead marking. Our scheme almost completely eliminates the effect of legacy routers when they constitute less than 20% of the topology, and performs 2–4 times better than the original Pi scheme [40]. For the filtering mechanism, we derive an optimal threshold strategy for endhosts and edge servers for filtering based on the Pi marking. We also develop a new filter, the *PiIP filter*, which can be used to detect IP spoofing attacks with a single attack packet. We also examine the conflicts between IPv4 fragmentation and Pi marking, and Pi deployment in an IPv6 environment.

The remainder of this article is organized as follows: In Section II we review the essential elements of the Pi scheme. In Section III we introduce our improvements to the Pi scheme, and evaluate them in Section IV. In Section V, we describe the PiIP filter, and in Section VI we discuss IP Fragmentation, IPv6 deployment and other issues. In Section VII we describe related work in the DDoS defense literature. Finally, we conclude in Section VIII.

II. Overview of Pi

In this section, we present the key design elements of the Path Identification (Pi) scheme. A full description of Pi can be found in an earlier work [40].

II-A. Pi Properties

The Pi DDoS defense scheme is composed of a packet marking algorithm that encodes a complete *Path Identifier* (Pi) in each packet; and a packet filtering algorithm, that determines how a DDoS victim will use the markings of the packets it receives to identify and filter attack packets. The uniqueness of Pi lies in the fact that the Pi marking scheme is deterministic at the path level: all packets traversing the same path receive the same marking. Because each packet contains the complete path marking, and the marking for a path is unchanging, then the victim need only identify a single attack packet or flow (through some high level algorithm based on packet contents or flow behavior) in order to block all subsequent packets arriving from the same path, and presumably, from the same attacker. The next two sections describe the details of how Pi marking and filtering work.

II-B. The Pi Marking Scheme

The Pi marking scheme defines how the Pi-marks are generated as a packet traverses the routers along its path to its destination. Each Pi enabled router marks n bits into the IP Identification field of every packet it forwards — where n is a global constant equal to either 1 or 2. The IP Identification field is broken into $\lfloor 16/n \rfloor$ sections, and each router marks its n bits into the section indexed by the packet's current TTL modulo $\lfloor 16/n \rfloor$. Because the IP Identification field is 16 bits in length, each Pi-mark can hold markings from the last 8 ($n = 2$) or 16 ($n = 1$) routers away from the packet's destination — a new router marking simply overwrites the marking of a previous router.

Our research on Pi shows that the markings of the last 8 or 16 routers suffice for filtering out the majority of DDoS traffic, even though many different paths carry the same marking. Our analysis of the Internet map [7] and the Skitter traceroute maps [8] indicates that the average Internet path length is roughly 15, which is almost double the number of hops that the $n = 2$ bit scheme can hold. Thus, the victim receives the markings from only the last 8 routers in the $n = 2$ bit scheme. We find that the filtering power of Pi improves if we prevent the local domain routers from marking, thus preserving the markings from routers further away. For example, if the last 3 hops are routers within our domain, we assume that we can configure them not to mark packets destined for our domain. Internet packets would thus carry the markings from routers 4 to 11 hops away (assuming an $n = 2$ marking scheme).

It is critically important that the individual router's markings have as high an entropy as possible, so that the probability of two distinct paths sharing — or, colliding at — the same Pi marking is as small as possible. For this reason, the router's

marking bits are computed as the last n bits of the MD5 hash of the current router's IP address concatenated with the last-hop router's IP address. A Pi enabled router would cache its marking bits for each interface to avoid recalculating the hash for each forwarded packet.

The original Pi mark works well in a network where all routers implement Pi marking. Unfortunately, performance degrades substantially if *legacy* routers are present, as they decrement the TTL but do not mark the packet. In this article, we introduce two new techniques that greatly enhance the performance of Pi in the presence of legacy routers: the Stack marking and the Write Ahead improvement, which we describe in Section III.

II-C. The Pi Filtering Scheme

The Pi filtering scheme defines how a DDoS victim uses the Pi-marks of the packets it receives to accept the least amount of attack traffic while accepting the most amount of legitimate traffic. The simplest Pi filtering scheme is as follows: upon identifying a particular Pi-mark as belonging to an attacker (by observing malicious behavior in a packet or flow of packets sharing a Pi mark), the victim drops *all* subsequent packets bearing the same Pi-mark. Unfortunately, because there are a constant number of Pi marks (2^{16}), as the number of attackers increases it is more and more likely that any given Pi mark will receive some attack packets, hence causing all legitimate user traffic to be dropped as well. This effect is called *marking saturation*.

To cope with *marking saturation*, the victim needs to have more flexibility in deciding whether or not to reject all packets with a particular Pi-mark. This flexibility can be defined in terms of a *threshold*: a value measured as the maximum allowable ratio of attack packets bearing a particular Pi mark to the total number of packets arriving with that Pi mark. In a threshold filter, the victim will only drop all packets with a particular Pi-mark if the ratio of attack to total traffic on that Pi-mark equals or exceeds the threshold value.

III. StackPi: A New Marking Scheme for Pi

The Pi marking scheme presented in the previous section performs well under the idealistic assumption that all routers in the Internet implement it. However, one of the criterion for a DDoS defense presented in Section I-B was that the scheme support incremental deployment; where not all routers in the Internet participate in the marking algorithm. In this section, we first explain the weakness of the original Pi marking algorithm (referred to as the *TTL marking* algorithm), and then present two new schemes, *Stack marking* and router *write-ahead*, which both dramatically improve Pi's incremental deployment performance.

III-A. Incremental Deployment Issues

An important property in the Pi marking scheme is that all packets traversing the same path produce a single marking. This property relies on there being enough routers in a given path to completely overwrite the IP Identification field with their markings. Because the IP Identification field is initialized by the end-hosts (the attackers in the case of a DDoS attack), any bits that are not overwritten by the routers in the path can be used by an attacker to change between different markings. In the ideal scenario of 100% deployment of the marking scheme, this effect is limited only to short paths. However, as the percentage of non- marking (legacy) routers increases, the likelihood of completely overwriting the IP Identification field correspondingly diminishes. In fact, TTL marking is particularly vulnerable to this effect, because of the fact that legacy routers decrement the TTLs of the packets that they forward. Thus, a single legacy router can cause a section of the IP Identification field to go unmarked, at least until the TTL pointer wraps around again. However, with an $n = 2$ bit scheme, this requires 8 more hops to be present in the path, the 8th of which *must* not be a legacy router. These unmarked sections are called *marking holes* because they often go unfilled and leave attacker initialized bits in the Pi marking. We show the impact of legacy routers on TTL marking in Section IV.

III-B. Stack Marking

The intuition behind Stack marking is the same as that for TTL marking: in order to generate a path identifier that is representative of a particular path from a source to a destination in the Internet, each router along the path must contribute some small amount of information whose aggregate among the routers of the path will be the Pi marking. However, instead of using the packet's TTL to aggregate the markings from different routers, each router instead treats the IP Identification field as though it were a stack.[1] Upon receipt of a packet, a router shifts the IP Identification field of the packet to the left by n bits and writes its marking bits (calculated in the same way as in TTL marking) into the least significant bits that were cleared by the shifting (as shown in Figure 1). In other words, the router simply *pushes* its marking onto the stack. Because of the finite size of the Identification field, the n most significant bits, which represent the oldest mark in the packet, are lost in this process; just as in TTL marking. In fact, Stack marking and TTL marking are equivalent in the case of 100% deployment.

The differences between TTL and Stack marking become evident when legacy routers are introduced into the topology. Unlike TTL marking, which interacts poorly with legacy routers because of its reliance on the packet's TTL which is modified by legacy routers, Stack marking does not rely on the TTL, and hence, has no interaction with legacy routers at all. There are no longer any *marking holes* because each marking router places its mark adjacent to the last marking router's mark, in the least significant bits of the IP Identification field. Completely marking the whole field using Stack marking requires only that there be $\lfloor 16/n \rfloor$ non-legacy routers anywhere in the path.

[1] To be precise, the StackPi markings are treated as a *windowed stack, since the oldest markings are displaced by new markings, but* push and *pop operations still manipulate the most recently added markings.*

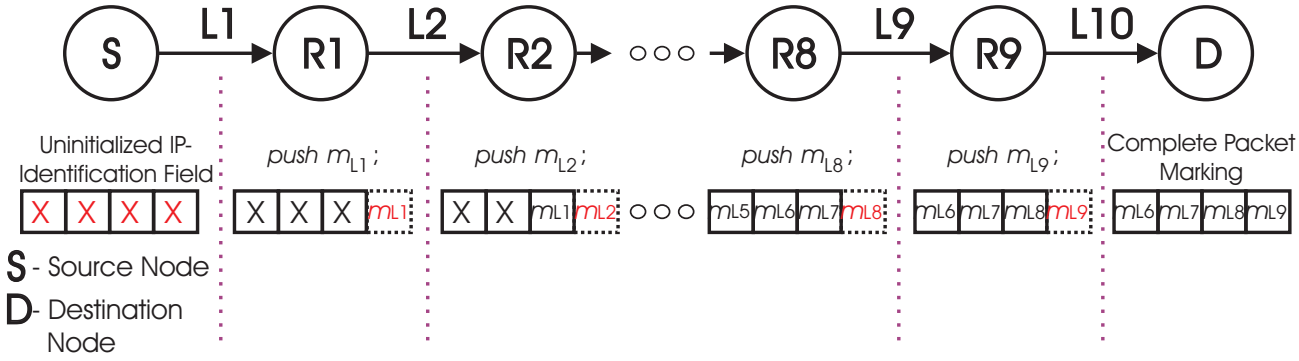

Fig. 1. The basic Stack Marking Scheme. This figure shows how the Pi mark evolves as the packet traverses routers R1 through R9. Initially, the marking field contains arbitrary data. In this example, each router marks with $n = 2$ bits and the field has space for four router markings.

III-C. Router Write-Ahead

Stack marking eliminates the interaction between Pi-enabled routers and legacy routers that is present in TTL marking. However, Stack marking is still limited in that a path which has too few marking routers will still result in end-host initialized bits arriving at the victim; which allows attackers to shift between different Pi markings. We can add an extra step to our marking scheme to improve this situation slightly. We already assume that each router knows the IP address of the last-hop routers or hosts from which it receives packets (this knowledge is necessary for generating the marking bits of each router, as explained in Section II-B). If we also assume that each router knows the IP address of the next-hop routers or hosts to which it is forwarding packets, then the router is capable of marking the packets *on their behalf*. All the router needs to do is substitute its own IP address for the last-hop IP address and the next-hop IP address for its IP address when calculating the bits to mark (of course, the results of this calculation should also be cached so that they need not be repeated for each forwarded packet). This second marking is called *Write-ahead marking*.

The benefits of Write-ahead marking are immediately evident when considering a Pi-enabled router followed by a legacy router in a path. In this case, the Pi-enabled router will mark not only for itself but also for the next-hop router, the legacy router, so that its marks will be included as well.

There is, however, a slight complication with the Write-ahead scheme: what happens when two Pi-enabled routers are adjacent to each other in a path? It would be a waste of space for the second Pi enabled router to add its own mark to the packet, since the first Pi-enabled router would have added that mark already. Therefore, we are forced to change our scheme slightly to accommodate this situation. Upon receipt of a packet, a router *peeks* (the process of looking at the item in the top of the stack without modifying the stack itself) at the least significant n bits and compares the marking it finds there to what its own marking would be. If the markings are identical, then the router will assume that the last hop router is Pi-enabled and has already marked the field on the current router's behalf. In this case, the current router will skip pushing its own marking and will only push the next-hop router's marking onto the stack. If the topmost marking is different from what the router's marking would be, then the router will assume that the last-hop router was a legacy router and will push its own marking as well as the next-hop router's marking onto the stack. There is a chance that a legacy router placed between two Pi-enabled routers will go undetected if it has the same marking as the Pi-enabled router after it; this probability is equal to $\frac{1}{2^n}$ where n is the number of bits in each router's marking. Figure 2 shows an example of the stack based scheme with write ahead. However, even if a legacy router is missed in the Write-ahead scheme, there is no wasted space in the IP Identification field; it is simply as though the Write-ahead scheme was not used at all. Therefore, the Stack marking scheme with Write-ahead is a strict improvement over the stack marking scheme alone.

It is possible to build a mechanism that would allow a router to detect that its last-hop neighbor in a path is a legacy router. This mechanism would allow a router to simply push its own markings onto the stack without incurring the $\frac{1}{2^n}$ probability that the last Pi-enabled router's marking is identical to the current router's marking (causing the current router not to add its own markings, as discussed above). Such a mechanism could be as simple as noting a variation in the least-significant bits of the incoming Pi mark between packets arriving on the same interface and from the same last-hop (as determined by the Link Layer address). This mechanism works because a non-legacy (Pi-enabled) router would *always* mark the same bits into the packet, resulting in the same bits arriving to the next hop router. Thus, a variation in the least significant bits observed in the Pi-mark of a specific last-hop router indicates that at least that last-hop router is a legacy router. Unlike the Write-ahead scheme proposed above, this scheme may result in extra markings being added to the stack in the particular scenario where there exists a layer-2 network between three or more Pi-enabled routers. The layer-2 routers will rewrite the Link Layer addresses of packets, which may cause one Pi-enabled router to see different Pi marks (generated by the other two Pi-enabled routers) appear on the same interface with the same Link Layer address. To avoid the uncertainties with this legacy router detection scheme, we use the standard write-ahead scheme presented above in our subsequent simulations.

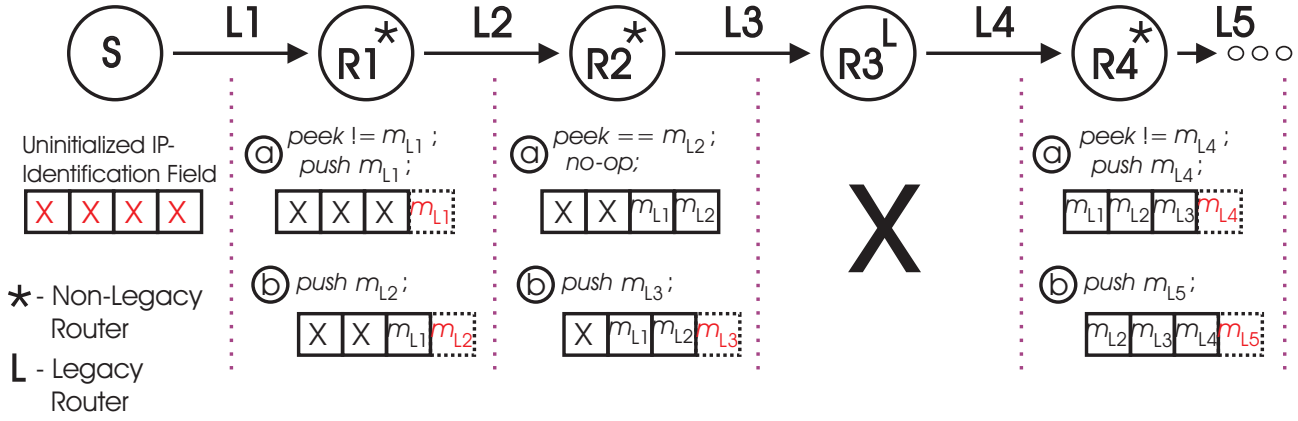

Fig. 2. The stack marking scheme with write-ahead. The new scheme allows the inclusion of markings from router R3, despite the fact that it is a legacy router. Each router along the path first (a) checks the topmost marking in the stack to see if it equals the marking that would have been generated by the router connected to the current router's incoming link, and if the topmost marking is not equal to that, the router adds that marking to the packet; and then (b) adds the marking for the incoming link of the next-hop router to the stack.

IV. Analysis and Evaluation of the StackPi Improvements

In this section, we evaluate the performance of the Pi scheme under a simulated DDoS attack. We first review the DDoS attack model that we use in our simulations. We then derive an equation for the optimal value of the Pi *threshold filter* under attacks of varying severity and evaluate its performance. Finally, we evaluate the effect of the StackPi marking improvements on Pi's incremental deployment performance.

IV-A. DDoS Attack Model

In order to model Pi's performance under a DDoS attack, we must have some way for the DDoS victim to identify attack packets, so that it can bootstrap the Pi filter. Unfortunately, this requires the simulation of a higher-level algorithm that is likely to be dependent on the content of the traffic (HTTP or DNS etc.) to make its classifications. Simulating such an algorithm is beyond the scope of this article.

To compensate for this, we model our DDoS attack in two phases: the *learning phase* and the *attack phase*. In the learning phase, the victim is considered omniscient, and can determine, for each packet received, whether that packet originated from an attack or a legitimate user. This phase of the attack is used to simulate the effect of a high-level traffic and content analysis algorithm, without specifying the algorithm itself. The knowledge gained in the learning phase is used to bootstrap the Pi filter with the Pi markings of known attackers. In the attack phase, the victim can no longer differentiate attack and user packets and is forced to use the Pi filter to make accept or drop decisions for every packet it receives. All of the results presented are taken during the attack phase. The length of the learning phase is 3 packets per legitimate user and 30 packets per attacker. The length of the attack phase is 20 packets per legitimate user and 200 packets per attacker.

For our experiments, we use Burch and Cheswick's Internet Mapping Project [6], [16] topology and the Skitter Project topology distributed by Caida [8].[2]

[2]Due to space limitations, we only show the results from the Skitter Map topology.

Our DDoS simulations proceed as follows: a certain number of *paths* are selected, at random, from the topology file and assigned to be either attack or legitimate user paths. All of the DDoS simulations have 5,000 legitimate users and vary the number of attackers. We use an $n = 2$ bit marking scheme and assume, as discussed in Section II-B, that the last three hops of any path are under the victim's ISP control and thus, do not add their marks to the packet. The results presented are the averages of 6 runs of each attack.

IV-B. Threshold Filtering Performance in StackPi

Recall from Section II-C that the threshold value of the Pi filter is used to give a DDoS victim some flexibility in deciding whether or not to drop all packets arriving with a particular Pi mark by setting a minimum acceptable level of user traffic to that Pi mark. We showed in earlier work [40] that the greater the severity of the attack, the better higher threshold values performed. In this section, we derive the formula for the optimal threshold value as a function of attack and user traffic, and confirm the optimality of our result using our DDoS simulation.

In order to quantify the performance of the Pi filter, we first define two metrics, representing the two different types of errors a Pi filter can make: *false positives*, where legitimate users' packets are dropped; and *false negatives*, where attackers' packets are accepted. For the purpose of our evaluation, we refer to the following two metrics: the *user acceptance ratio*; which is 1 minus the false positive rate, and the *attacker acceptance ratio*; which is exactly the false negative rate. We define these two metrics in terms of the following simulation variables:

p_j - The total number of packets sent by entity j

v_j - The total number of packets sent by j accepted by the victim.

The acceptance ratio, a_j, for a given entity j is defined as:

$a_j = \frac{v_j}{p_j}$

Thus, for the set of all users, U, and all attackers, A, the acceptance ratios are defined as $a_U = \frac{v_U}{p_U}$ and $a_A = \frac{v_A}{p_A}$ for the users and attackers, respectively.

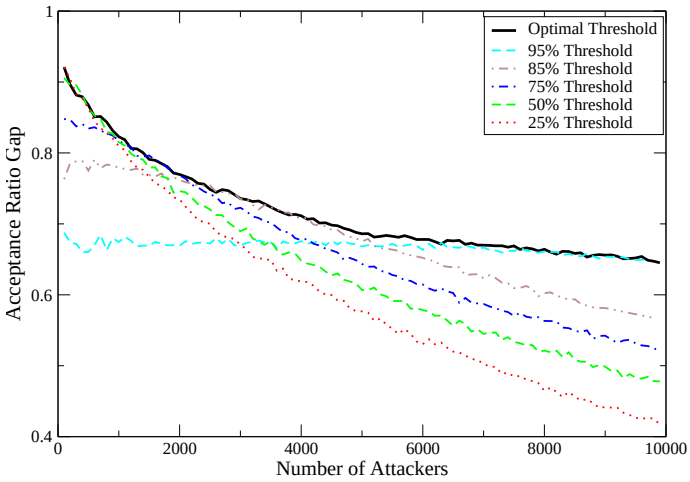

Fig. 3. Performance Comparison of threshold values.

Under a DDoS attack, the victim would like to maximize the user acceptance ratio and minimize the attacker acceptance ratio. The two acceptance ratios are correlated in threshold filtering schemes, as a decrease in one can result in the increase of the other. The goal of the victim, then, is to maximize the difference between the two ratios:

$$\Delta = \frac{v_U}{p_U} - \frac{v_A}{p_A}$$

which we refer to as, Δ, the *acceptance ratio gap*. The acceptance ratio gap is a useful metric to determine how a particular StackPi filter performs relative to other filters or to no filter at all. Without Pi marking and filtering, the victim can only make accept/drop decisions at random, which intuitively gives an equal user and attacker acceptance ratio, or an acceptance ratio gap of zero. Relative to other filters, a better filter has a higher acceptance ratio gap.

In order to maximize the acceptance ratio gap, we must find an optimal threshold value. We derived the optimal threshold value as follows:

$$f_i = \begin{cases} 1 & \text{if } \frac{p_{U_i}}{p_{A_i}} < \frac{p_U}{p_A}, \\ 0 & \text{otherwise.} \end{cases}$$

$$t_{opt} = \frac{p_U}{p_A}$$

The details of the derivation are in the Appendix.

This result indicates that in order to maximize the acceptance ratio gap, unless the ratio of user traffic to attack traffic at a particular Pi mark is greater than the ratio of user traffic to attack traffic *over all Pi marks*, then all packets bearing that marking should be dropped. Because we prefer to deal with thresholds as percentages, we normalize our optimal threshold value to be:

$$t_{opt} = \frac{p_U}{p_A + p_U}$$

To calculate the value of the threshold the victim uses the information from packets in the learning phase of the DDoS attack to set the value of the threshold in the attack phase of the DDoS attack. Figure 3 shows the performance of the optimal threshold filter relative to select constant threshold values for attacks of increasing severity. Each of the constant threshold value curves is tangential to the curve of the optimal threshold and intersecting at a single point on the curve where the optimal value of the threshold equals the value of the constant threshold.

IV-C. Legacy Router Analysis

We now apply the optimal threshold filter to the two marking schemes: the TTL marking scheme from the earlier Pi work [40] and the StackPi marking scheme introduced in Section III. Figures 4 and 5 show the acceptance ratio gaps for the TTL and StackPi marking schemes, respectively, under increasing percentages of legacy routers.[3]

The TTL marking scheme performs as expected (and similarly to its performance in [40]), with a roughly constant decrease in performance per added percent of legacy routers. This confirms our assertion from Section III-A that *marking holes* generated by legacy routers go mostly unfilled, and result in more Pi markings per attacker and hence, less filtering accuracy at the victim.

The situation is much improved using the StackPi marking scheme. The slow performance degradation at low percentages of legacy routers is due to two phenomenon. The first is the elimination of the marking holes due to the stack based marking. Because most paths contain more routers than there is space for in the IP Identification field, when some routers stop marking, other routers' marks simply take their place. The second phenomenon is the effect of the write-ahead improvement. At low percentages of legacy routers, it is likely that a legacy router will appear between two Pi-enabled routers. In this case, the write-ahead improvement allows for that legacy router's markings to be included by the Pi-enabled router appearing before it, so the legacy router has no effect on the Pi mark for that path. As the percentage of legacy routers goes beyond 60%, these two effects are minimal, and the performance degradation per percent increase in legacy routers is equivalent between the TTL and StackPi marking schemes.

Overall, the StackPi marking scheme outperforms the TTL marking scheme at all percentages of legacy routers, particularly the low percentages. With StackPi marking, the Pi scheme provides some DDoS protection, even when as little as 20% of routers in the Internet implement the scheme.

[3]Unfortunately, our algorithm for introducing legacy routers into the topology results in a uniform distribution, which may be unrealistic. It is more likely that clumps of routers in a path - perhaps belonging to a particular organization - will be updated over a short period of time. A uniform distribution also biases the results in favor of the StackPi marking scheme because it is more likely that a single legacy router, rather than a series of them, will appear in a uniform distribution. We are working on experiments that utilize different distributions to model the occurrence of legacy routers in the network topology.

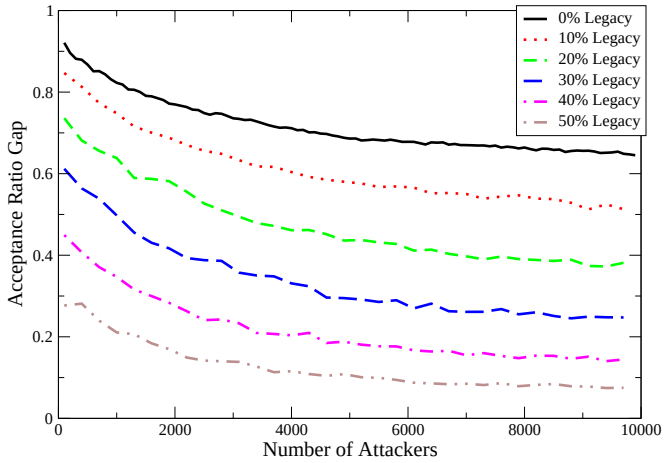

Fig. 4. TTL marking incremental deployment performance.

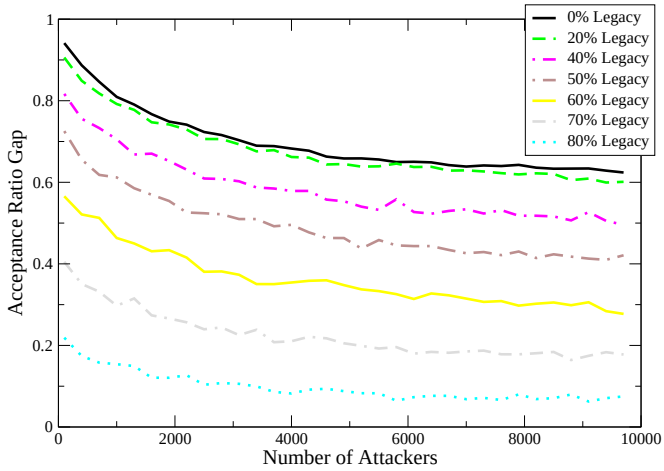

Fig. 5. StackPi marking incremental deployment performance.

V. PiIP Filter: Filtering on the < Pi Mark, Source IP Address > Tuple

The optimal threshold filtering we describe in Section IV-B is a great general filtering technique with low filtering overhead. In this section, we present a more powerful filter, the PiIP filter, which can detect IP spoofing attacks on a per-packet basis. Recent work has shown that IP spoofing is possible in at least a quarter of the ASes of the Internet [4] and this remains a useful vector for attackers launching DDoS attacks. In this section, we limit our focus to spoofing attacks where the attacker tries to spoof an IP address that is not from the same network as the attacker itself.

V-A. PiIP Filter Design

The key observation is that assuming relatively stable Internet forwarding paths, packets originating from an IP network will arrive at the destination with a small set of distinct Pi marks. Conversely, assuming that the Pi marks are approximately uniformly distributed, a given Pi mark will only be produced by a relatively small number of origin networks. These assumptions hold in particular for networks where packets traverse sufficiently many Pi routers such that all bits in the IP identification field are marked.

We observe that packets from a given IP network will all arrive at the destination with a small number of distinct Pi marks — we can use this to design a powerful filter to reject packets with spoofed IP addresses. We consider the following setup. During peace time (when a server is not under attack), the server stores the tuple < Pi mark, source IP address >, or <Pi,IP> for short.[4]

When the server is under attack, it uses the <Pi,IP> database to filter out packets with spoofed source IP addresses. For each incoming packet, the server checks whether the <Pi,IP> tuple of the arriving packet matches an entry in the database; if the tuple does not match the corresponding entry in the database, it rejects the packet. A nice feature of this PiIP filter is that the server can filter out the very first malicious attacker packet. However, the forwarding path of a legitimate receiver may change and the arriving packet's <Pi,IP> tuple may not be in the database. Thus, the application writer needs to consider the output of the PiIP filter as a *hint* on whether the source IP address is spoofed or not. As long as the server has sufficient capacity, questionable packets may also get served, and if the packet originator turns out to be a legitimate user, the server can add the <Pi,IP> tuple to its database. Note that the PiIP filter cannot be used to detect IP spoofing attacks if the IP address in the packet is not in the database. We have several ways to address this issue. Because packets from the same network (even if not from the same IP addresses) usually have the same Pi mark, from the Pi mark of one IP address we can derive the Pi mark of other IP addresses on the same network.

V-B. PiIP Filter Evaluation

To evaluate the performance of the PiIP filter, we compute the probability of a false negative, i.e., the probability an attacker can send a packet with a spoofed IP address that the victim accepts. A false negative occurs if the attacker spoofs the IP address of an end-host that happens to have the same Pi mark as the attacker itself. It is clear that the PiIP filter performs best if a given IP address has very few possible Pi marks, and if Pi marks are well distributed. For example, assuming uniformly distributed Pi marks and assuming that a given IP address has 4 possible Pi marks for a certain destination, an attacker has a $4/2^{16} = 1/2^{14}$ probability to spoof the IP address to that destination so that its packet will be accepted.

We conduct the following experiment: each end-host sends 10 packets with non-spoofed source IP addresses to the victim to build the <Pi,IP> tuple database. Figure 6 shows a histogram of the number of Pi markings with a particular number of unique IP addresses that map to them (note that the y-axis is logarithmic). The histogram shows us that the IP addresses are somewhat uniformly distributed over the possible Pi marks, with the large majority of Pi marks having 1 to 4

[4]Since in general all packets from a network have the same Pi marks, the server can store the network address instead of the source IP address. However, for simplicity we discuss the case where the server stores the IP address.

unique IP addresses that map to them and very few Pi marks with greater than 20 unique IP addresses that map to them.

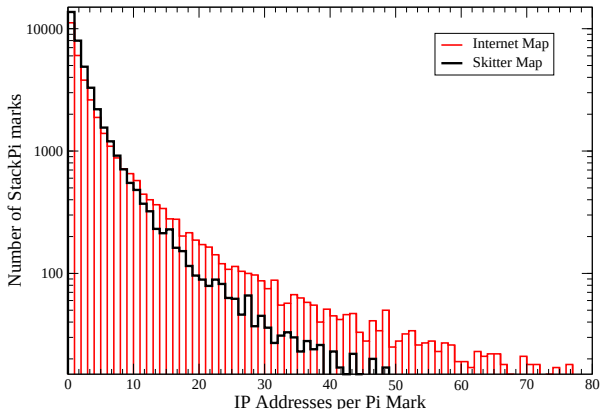

Fig. 6. Histogram of the frequency of Pi marks with a particular number of IP addresses that map to them, after 10 packets are sent from each end-host in the Internet Map topology.

Given this distribution, we can calculate the probability that a randomly chosen end-host from the topology will succeed in spoofing the IP address of another end-host. Because we are dealing with a topology that only contains a small subset of all the possible legitimate IP addresses, we assume that our attacker has access to a list of the unique IP addresses of all end hosts in our topology and selects addresses from this list when spoofing a packet. We begin by calculating the probability that an attacker with a particular IP address, k, will succeed in spoofing a packet that will be accepted by the filter. This probability depends on how many Pi markings have been recorded at the victim for address k. We define the set of n distinct Pi markings recorded for address k as $\{m_0, m_1, \ldots, m_n\}$. For each Pi mark recorded at the victim for IP address k, there is a set of other IP addresses that also map to the same Pi mark. If the attacker were to spoof any of these, the attack packet would be accepted by the filter. Thus, the probability of an attacker with IP address k successfully spoofing is:

$$P_k = \frac{\sum_{i=0}^{n} uniqueIPs(m_i, k)}{N}$$

where the $uniqueIPs()$ function returns the number of unique IP addresses that map to Pi mark m_i, excluding IP address k as well as any duplicates between function calls, and N represents the number of end-hosts in the topology; which is the size of the list of possible IP addresses that the attacker can spoof. Given the probability of an attacker with a specific IP address of successfully spoofing a packet, we can now calculate the probability of an attacker with a random IP address successfully spoofing:

$$P = \frac{\sum_{k=0}^{N} P_k}{N}$$

Using the values from the 10 packet bootstrapping experiment, we calculated this probability to be 0.005 for the Internet Map topology and 0.003 percent for the Skitter topology. Although this result is two orders of magnitude worse than the ideal case, it still shows that in real topologies an attacker has a very small chance to successfully spoof another IP address that is not from the same network as the attacker.

VI. Discussion

The Pi scheme has great potential as a DDoS and IP spoofing defense mechanism. In this section, we discuss a number of issues and extensions relating to Pi that could substantially improve on the results we have obtained thus far. In Section VI-A we discuss an extension to the StackPi marking scheme that allows routers to mark with a variable number of bits. In Section VI-B we show how Pi-IP filters can be used to implement a form of standard IP traceback. In Section VI-C we discuss Pi's compatibility with IP fragmentation. In Section VI-D we explain briefly how Pi can be applied in an IPv6 environment. Finally, in Section VI-E we discuss why the Pi scheme is fundamentally different from other IP traceback schemes from a deployment perspective.

VI-A. Variable Bit Marking

Thus far, we have assumed that the Pi marking scheme has a global parameter n, which is the number of bits that each router uses to mark a single link. We may gain more space efficiency if routers are allowed to choose for themselves a particular n to use for marking. For example, in the current Pi marking scheme, a router with only two interfaces would mark two bits in the packet, although that router does not truly affect the path at all, since it can be abstracted simply as a link between its last-hop and next-hop neighbors. In a variable bit marking scheme, such a router would not mark the packet. Each router would calculate its own n, possibly as a function of the number of interfaces it has. We are working on simulations that incorporate the variable bit marking scheme into Pi.

VI-B. Enabling Traceback with PiIP filters

A properly bootstrapped PiIP filter (as described in Section V) can be used to perform standard traceback, that is, complete path reconstruction from a packet's destination to its sender. When a destination receives a packet that is flagged because its source IP address does not match its Pi marking in the Pi-IP filter's database, the victim can consult the database to generate a list of IP addresses that correspond to the packet's Pi mark. The victim can then determine the path by simply executing `traceroute` and recording the path.[5] Although this method does not guarantee a unique path to the packet's origin (there may be multiple IP addresses that map to the same Pi mark), it does greatly reduce the space of potential attackers.

VI-C. Compatibility with IP Fragmentation

Placing a deterministic marking in the IP Identification field of every packet in the network is incompatible with the current IPv4 fragmentation mechanism (except under very strict network assumptions such as no packet reordering or loss).

[5]This process will not accurately reconstruct asymmetric paths.

Despite the fact that fragmented traffic represents between 0.25% and 0.75% of packets in the Internet [36] [33], we must at least consider a mechanism to allow Pi to coexist with fragmentation.

We offer a solution, proposed by Vern Paxson, that routers only mark packets that will never get fragmented and that are not fragments themselves. The latter class is simple to identify, as these packets will have a non-zero Fragment Offset field in their header or a `more fragments` flag which is set. Determining which packets will never get fragmented is more challenging. The simplest classification is those IP packets that have the `Do Not Fragment` (`DF`) bit set in the Flags field of the IP header. This classification is adequate for servers with a majority of TCP traffic – as most modern TCP implementations set the `DF` bit by default [39], as specified by the Path MTU Discovery standard in RFC 1191 [27]. During a DDoS attack, a server can easily filter out packets that do not match this classification and are thus not marked. We show the percent of markable traffic from a 31 day trace of packets from the Lawrence Berkeley Lab DMZ in Table I. From Table I, we see that if an attacker intentionally does not set the `DF` bit to evade marking, filtering out those packets at the victim will only adversely affect 1.76% of legitimate traffic.

Unfortunately, the `DF` classification is inadequate for UDP traffic, which has a much smaller percent of traffic that carries the `DF` bit. Without the `DF` bit, classifying packets that will never be fragmented is no longer 100% accurate. An alternative method would be to only mark UDP traffic that is smaller than the smallest Maximum Transmission Unit (MTU) for common Internet traffic links. A widely accepted value for this is 576 bytes [5], however, lower MTU links are possible and perhaps likely, with the expected proliferation of web-enabled phones. In either case, the networking community will need to agree on a specific value before Pi can be deployed to protect UDP specific services.

Packet Classification	Percent markable
TCP with DF	98.24%
UDP with DF	26.69%
UDP $\leq$576b or DF Set	87.12%
UDP $\leq$250b or DF Set	79.06%
UDP $\leq$100b or DF Set	64.75%

TABLE I

PERCENT OF PACKETS THAT CAN BE MARKED BY CLASSIFICATION. AVERAGE OVER 31 DAYS OF TRAFFIC FROM LAWRENCE BERKELEY LAB DMZ, MAY 1-31, 2003.

VI-D. StackPi in IPv6

Although the Pi scheme has been specifically designed for deployment in IPv4, its principal ideas are equally applicable in an IPv6 environment. The IPv6 protocol does not support en-route packet fragmentation, and thus does not have an equivalent field to the IP Identification field of IPv4. There are, however, two possibilities for marking space in IPv6: in the *flow identification* field or in a *hop-by-hop* option. The advantage of marking in the flow identification field of the header is that because the field is part of the standard header, router markings will not add to the packet's size (which might cause the packet to exceed the MTU of an intermediate network and be dropped). The flow identification field is 20 bits in length, which allows more routers to include their markings in each Pi-mark.[6] Of course, this is not the purpose that the flow identification field was meant to serve [12].

The other option is to include the Pi marking in a hop-by-hop option inserted by the first Pi enabled router in the path. The benefit of this approach is that the length of the option need not be limited to 20 bits, as is the flow identification field. However, inserting such an option into the packet may cause it to exceed the MTU of a link somewhere along the path. In either case, DDoS protection is a critical feature that should be present at the network level, and IPv6's current limited deployment makes it a good candidate for modification to include the Pi scheme.

VI-E. Incentives for Deployment

Previous DDoS defense mechanisms do not provide a good incentive structure to foster adoption. For example, consider the benefits to an ISP deploying ingress filtering [13]. That ISP protects other ISPs' customers from its own customers, as ingress filtering stops its customers from spoofing their source IP address. Ingress filtering does not directly benefit the customers of the ISP, yet it introduces more complexity, higher router management overhead, lower performance due to filtering, and potential customer problems (when some legitimate customer's packets get filtered out).

In contrast, the Pi scheme offers very good incentives for deployment that encourage adoption. First of all, if an ISP deploys Pi marking on all its routers, a customer can immediately start using the filtering techniques we describe in this article to determine from where the attack traffic enters its ISP's topology. As we demonstrate in Section IV-C, a victim can already perform filtering if only 20% of the routers implement Pi marking. Ideally, this creates a market pressure for ISPs to deploy Pi enabled routers. If ISPs want to deploy Pi, this creates an incentive for router manufacturers to produce Pi-enabled routers. We anticipate that the benefits of Pi will produce these market incentives that drive deployment. The main difference with previous techniques is that Pi deployment immediately benefits the customers of an ISP, and helps those customers defend against DDoS attacks.

VI-F. Changing Pi Marks

One of the basic assumptions of the Pi scheme is that the paths from specific senders remain constant over the timescale of an attack. Attackers can exploit this assumption in a variety of ways. Instead of focusing a DDoS traffic flood on a particular victim, an attacker can try and flood the routers along the path to the victim, potentially causing a disruption in the paths packets take to reach the victim, resulting in new Pi marks arriving at the victim. A clever victim may be able to identify the router under attack by comparing the Pi marks of traffic

[6]Unlike other IP Traceback schemes, which reconstruct the IP addresses of the routers in the path to the attackers, the fact that IPv6 addresses are 128 bits instead of 32 bits is completely transparent to the Pi scheme.

before and after the attack begins. Colluding attackers may try to poison the PiIP filter by coordinating to complete a TCP connection, while spoofing an address belonging to a single attacker. The end-host may be fooled into including the Pi marks of all attackers as legitimate Pi marks of the one attackers IP address. This attack is limited, however, because the attackers would need to spoof the same address, or set of addresses, (if they poison more than one attackers address) during the flooding phase of their attack.

VII. Related Work

There have been several studies of the frequency and nature of Internet DoS attacks [14], [15], [20], [28].

Many approaches have been proposed for securing against DoS and DDoS attacks. Ferguson and Senie propose to deploy network ingress filtering to limit spoofing of the source IP address [13]. However, unless every ISP implements this scheme, there will still be entry points in the Internet where spoofing can occur. Also, the additional router configuration and processing overhead to perform this filtering is another reason why it may not be widely deployed. Stone suggests a mechanism whereby ISPs use routers capable of input debugging connected through IP tunnels to an ASes border routers to enable AS-level tracing [37].

Park and Lee propose a distributed packet filtering (DPF) mechanism against IP address spoofing [29]. DPF relies on BGP routing information to detect spoofed IP addresses. Their approach is interesting, but requires high levels of router participation.

Bellovin et al. suggest adding a new type of ICMP message for traceback [3], [18], and Mankin et al. present an improvement to this scheme [26]. Several researchers propose to embed traceback information within the IP packet [1], [7], [11], [23], [31], [35], [42]. Most of these schemes use the 16-bit IP Identification field to hold traceback information. Routers along the packet's path probabilistically mark certain bits in the IP Identification field in certain ways. While the traceback schemes could be used to find the origins of the attacks, they often require a large number of packets and thus cannot be used to filter out packets on a per-packet basis.

Snoeren et al. propose using router state to track the path of a single packet [34]. 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 then recursively query their upstream routers for the presence of the packet in their hash tables. This mechanism works well if all routers deploy this approach, but requires routers to store substantial amounts of state and requires the victim to contact the routers for traceback.

Ioannidis and Bellovin, and Mahajan et al. propose Pushback, a packet filtering infrastructure leveraging router support to filter out DDoS streams [17], [25]. Jin, Wang and Shin propose the use of packet TTL as an effective means of identifying spoofed traffic [19]. The mechanisms we propose in this article can be used to greatly increase the effectiveness of Pushback and Hop-count filtering, as the filters can take the packet markings into account and thus distinguish packets from various origins (increasing the accuracy of filtering).

Sung and Xu propose an altered IP traceback approach, where the victim tries to reconstruct the attack path but also attempts to estimate if a new packet lies on the attack path or not [38]. Their scheme is probabilistic and each router either inserts an edge marking for the IP traceback scheme or a router marking identifying the router. Unfortunately, their approach requires the victim to collect on the order of 10^5 attack packets to reconstruct a path, and once the path is reconstructed, this scheme will likely have a high false positive rate as the routers close to the victim will all lie on some attack path and frequently mark legitimate packets which will then get rejected.

We have recently proposed a marking scheme *Pi*, a Path Identification algorithm [40]. The original Pi marking is based on the use of the packet's TTL field as an index into the IP Identification field where a router should add its marks. This method is not as lightweight as the StackPi method. Legacy routers have a harmful affect on the original Pi scheme because they decrement the TTL of a packet but do not add any markings. The StackPi scheme is robust to legacy routers and even includes the write-ahead scheme to incorporate markings for single legacy routers in the path.

Collins and Reiter use a novel approach of combining Cisco NetFlow data from a large network with Skitter map data, to compare DDoS defense mechanisms [10]. They measure the effectiveness of path aware defense systems (Pi and Hop-Count Filtering), as well as Static and Network-aware clustering.

Recently, network capability-based systems have been proposed for DDoS defense. Machiraju et al. propose a secure Quality-of-Service (QoS) architecture that is based on network capabilities [24]. Lakshminarayanan et al. leverage the i^3 infrastructure to enable a receiver to cut off unwanted senders [22]. Anderson et al. [2] present an infrastructure where the sender uses a capability to set up a path to the receiver. We subsequently proposed SIFF, a capability-based system that allows a receiver to enforce flow-based admission control [41]. Yang et al. propose a capability-based mechanism with fine-grained service levels that attempts to address the denial-of-capability attack [43]. They leverage Pi markings to filter out floods of request packets — in their scheme routers attempt to provide fair sharing among capability request packets based on their Pi markings. This nicely illustrates that Pi and StackPi are complementary to capability-based systems, and can be used to mitigate spoofing and flooding in the capability request channel.

VIII. Conclusion

In this article, we present new approaches for packet marking and filtering in the Pi DDoS defense scheme [40]. The StackPi marking improvements, stack-based and write-ahead marking, eliminate the *marking holes* generated by legacy routers and include the markings from single legacy routers immediately following Pi-enabled routers in a path. We derive an equation that allows a DDoS victim to select the optimal

threshold value for the Pi filter. We also introduce a novel filter which relies on the ⟨Pi, IP⟩ tuple of each packet, making it far less likely that an attacker will successfully bypass the filter. With these improvements, our evaluation shows that Pi provides measurable DDoS protection, even when only 20% of routers in the Internet participate in the marking scheme. Finally, we discuss how Pi can be made compatible with IPv4 fragmentation, and propose ways to integrate Pi marking into IPv6. The Pi scheme is very general and quite promising in performance. These properties promise to make Pi a critical deterrent to today's most common Internet attacks.

IX. Acknowledgments

We would like to thank Vern Paxson for his help with the research on making Pi coexist with IP fragmentation.

References

[1] Micah Adler. Tradeoffs in Probabilistic Packet Marking for IP Traceback. In *Proceedings of 34th ACM Symposium on Theory of Computing (STOC)*, pages 407–418, 2002.

[2] Tom Anderson, Timothy Roscoe, and David Wetherall. Preventing Internet Denial-of-Service with Capabilities. In *Proceedings of Hotnets-II*, pages 39–44, November 2003.

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

[4] Robert Beverly and Stephen Bauer. The Spoofer Project: Inferring the Extent of Source Address Filtering on the Internet. In *Proceedings of USENIX Steps to Reducing Unwanted Traffic on the Internet Workshop*, pages 53–59, July 2005.

[5] Robert Braden. Requirements for Internet Hosts – Communication Layers. Internet Request for Comment RFC 1122, Internet Engineering Task Force, October 1989.

[6] Hal Burch and Bill Cheswick. Internet Watch: Mapping the Internet. *Computer*, 32(4):97–98, April 1999.

[7] Hal Burch and Bill Cheswick. Tracing Anonymous Packets to Their Approximate Source. In *Proceedings of Usenix LISA*, pages 319–327, December 2000.

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

[9] CERT. TCP SYN Flooding and IP Spoofing Attacks. Advisory CA-96.21, September 1996.

[10] Michael Collins and Michael K. Reiter. An Empirical Analysis of Target-Resident DoS Filters. In *IEEE Symposium on Security and Privacy*, May 2004.

[11] Drew Dean, Matt Franklin, and Adam Stubblefield. An Algebraic Approach to IP Traceback. In *Network and Distributed System Security Symposium (NDSS)*, pages 1–10, February 2001.

[12] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. Internet Request for Comment RFC 2460, Internet Engineering Task Force, 1998.

[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] L. Garber. Denial-of-service attacks rip the Internet. In *IEEE Computer*, volume 33, April 2000.

[15] J. Howard. *An Analysis of Security Incidents on the Internet*. Ph.D. thesis, Carnegie Mellon University, August 1998.

[16] Internet Mapping. `http://research.lumeta.com/ches/map/`, 2002.

[17] John Ioannidis and Steven M. Bellovin. Implementing Pushback: Router-Based Defense Against DDoS Attacks. In *Proceedings of the Symposium on Network and Distributed Systems Security (NDSS 2002)*, San Diego, CA, February 2002.

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

[19] Cheng Jin, Haining Wang, and Kang G. Shin. Hop-count filtering: an effective defense against spoofed DDoS traffic. In *Proceedings of Conference on Computer and Communications Security*, pages 30–41, August 2003.

[20] Jaeyeon Jung, Balachander Krishnamurthy, and Michael 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.

[21] Mike Kristovich. Multi-vendor Game Server DDoS Vulnerability. `http://www.pivx.com/kristovich/adv/mk001/`, November 2002.

[22] K. Lakshminarayanan, D. Adkins, A. Perrig, and I. Stoica. Taming IP Packet Flooding Attacks. In *Proceedings of ACM HotNets-II*, pages 45–50, November 2003.

[23] Heejo Lee and Kihong Park. On the Effectiveness of Probabilistic Packet Marking for IP Traceback under Denial of Service Attack. In *Proceedings IEEE Infocomm 2001*, April 2001.

[24] S. Machiraju, M. Seshadri, and I. Stoica. A Scalable and Robust Solution for Bandwidth Allocation. In *International Workshop on QoS*, May 2002.

[25] Ratul Mahajan, Steven M. Bellovin, Sally Floyd, John Ioannidis, Vern Paxson, and Scott Shenker. Controlling High Bandwidth Aggregates in the Network. *CCR*, 32(3):62–73, July 2002.

[26] A. Mankin, D. Massey, C.L. Wu, S.F. Wu, and L. Zhang. On Design and Evaluation of Intention-Driven ICMP Traceback. In *Proceedings of the IEEE International Conference on Computer Communications and Networks*, October 2001.

[27] Jeffrey Mogul and Steve Deering. Path MTU Discovery. Internet Request for Comment RFC 1191, Internet Engineering Task Force, November 1990.

[28] David Moore, Geoffrey Voelker, and Stefan Savage. Inferring Internet Denial of Service Activity. In *Proceedings of the 10th USENIX Security Symposium*, Washington, D.C., August 2001.

[29] Kihong Park and Heejo Lee. On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets. In *ACM SIGCOMM '01*, pages 15–26, 2001.

[30] Vern Paxson. An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks. *Computer Communication Review*, 31(3):38–47, 2001.

[31] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson. Network Support for IP Traceback. *ACM/IEEE Transactions on Networking*, 9(3), June 2001.

[32] SETI@home. Search for Extraterrestrial Intelligence (SETI). `http://setiathome.ssl.berkeley.edu/`, 2003.

[33] Colleen Shannon, David Moore, and K. C. Claffy. Beyond folklore: observations on fragmented traffic. *IEEE/ACM Transactions on Networking (TON)*, 10(6), 2002.

[34] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Beverly Schwartz, Stephen T. Kent, and W. Timothy Strayer. Single-Packet IP Traceback. *IEEE/ACM Transactions on Networking (ToN)*, 10(6), December 2002.

[35] Dawn X. Song and Adrian Perrig. Advanced and Authenticated Marking Schemes for IP Traceback. In *Proceedings IEEE Infocomm 2001*, April 2001.

[36] Ion Stoica and Hui Zhang. Providing Guaranteed Services Without Per Flow Management. In *SIGCOMM'99*, pages 81–94, 1999.

[37] Robert Stone. CenterTrack: An IP Overlay Network for Tracking DoS Floods. In *Proceedings of the 9th USENIX Security Symposium*, pages 199–212, Denver, Colorado, August 2000.

[38] Minho Sung and Jun Xu. IP Traceback-based Intelligent Packet Filtering: A Novel Technique for Defending Against Internet DDoS Attacks. In *Proceedings of IEEE ICNP 2002*, November 2002.

[39] R. van den Berg and P. Dibowitz. Over-Zealous Security Administrators Are Breaking the Internet. In *Proceedings of USENIX LISA Conference*, pages 213–218, November 2002.

[40] Abraham Yaar, Adrian Perrig, and Dawn Song. Pi: A Path Identification Mechanism to Defend against DDoS Attacks. In *IEEE Symposium on Security and Privacy*, pages 93–107, May 2003.

[41] Avi Yaar, Adrian Perrig, and Dawn Song. SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 130–143, May 2004.

[42] Avi Yaar, Adrian Perrig, and Dawn Song. FIT: Fast Internet Traceback. In *Proceedings of IEEE Infocom*, March 2005.

[43] Xiaowei Yang, David Wetherall, and Tom Anderson. A DoS-limiting Network Architecture. In *Proceedings of ACM SIGCOMM*, pages 241–252, August 2005.

APPENDIX
OPTIMAL THRESHOLD FILTERING

We derive the optimal threshold filtering as follows.

Let v_{j_i} and p_{j_i} equal the number of packets accepted by the victim and the total number of packets sent by entity j, *with StackPi mark i*, respectively.

The acceptance ratio gap, Δ, is then:

$$\begin{aligned}\Delta &= \frac{v_U}{p_U} - \frac{v_A}{p_A} \\ &= \frac{\sum_{i=0}^{2^{16}-1} v_{U_i}}{p_U} - \frac{\sum_{i=0}^{2^{16}-1} v_{A_i}}{p_A} \\ &= \sum_{i=0}^{2^{16}-1} \left(\frac{v_{U_i}}{p_U} - \frac{v_{A_i}}{p_A} \right)\end{aligned}$$

When the ratio of user traffic to attacker traffic at a particular StackPi mark is above the threshold value, then all packets arriving with that StackPi mark are accepted. Thus, we can introduce our threshold filtering function, f_i, which will return 1 if the user traffic ratio at StackPi mark i is above the threshold and 0 if it is below. We can now define the packets accepted by the victim at a particular StackPi mark in terms of the total packets arriving at that mark:

$$v_{U_i} = f_i \cdot p_{U_i}$$
$$v_{A_i} = f_i \cdot p_{A_i}$$

We include these definitions in our acceptance gap equation:

$$\begin{aligned}\Delta &= \sum_{i=0}^{2^{16}-1} \left(\frac{f_i \cdot p_{U_i}}{p_U} - \frac{f_i \cdot p_{A_i}}{p_A} \right) \\ &= \sum_{i=0}^{2^{16}-1} f_i \left(\frac{p_{U_i}}{p_U} - \frac{p_{A_i}}{p_A} \right)\end{aligned}$$

To maximize Δ, it is clear that we must only accept packets with StackPi mark i where the ratio of user packets with StackPi mark i to the total number of user packets, $\frac{p_{U_i}}{p_U}$ is greater than the ratio of attack packets with StackPi mark i to the total number of attack packets, $\frac{p_{A_i}}{p_A}$. In terms of our filtering function f_i:

$$f_i = \begin{cases} 1 & \text{if } \frac{p_{U_i}}{p_U} < \frac{p_{A_i}}{p_A}, \\ 0 & \text{otherwise.} \end{cases}$$

Abraham Yaar worked on this project while he was a graduate student at Carnegie Mellon University. Avi obtained his MS degree in Summer 2004. He is now working at Microsoft and can be reached at ayaar@microsoft.com.

Adrian Perrig (M'96) is an Assistant Professor in Electrical and Computer Engineering, Engineering and Public Policy, and Computer Science at Carnegie Mellon University. He earned the Ph.D. degree in Computer Science in 2001 from Carnegie Mellon University, and spent three years during his Ph.D. degree at the University of California, Berkeley. He received the M.S. degree in Computer Science in 1999 from Carnegie Mellon University and the B.Sc. degree in Computer Engineering in 1997 from the Swiss Federal Institute of Technology in Lausanne (EPFL). Professor Perrig's research interests revolve around building secure systems and include Internet security, security for sensor networks and mobile applications.

Dawn Song is an Assistant Professor at Carnegie Mellon University. She obtained her PhD in Computer Science from UC Berkeley. Her research interest lies in security and privacy issues in computer systems and networks. She is the author of more than 35 research papers in areas ranging from software security, networking security, database security, distributed systems security, to applied cryptography. She is the recipient of various awards and grants including the NSF CAREER Award and the IBM Faculty Award.