SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks*

Abraham Yaar   Adrian Perrig   Dawn Song

Carnegie Mellon University

{ayaar, perrig, dawnsong}@cmu.edu

Abstract

One of the fundamental limitations of the Internet is the inability of a packet flow recipient to halt disruptive flows before they consume the recipient's network link resources. Critical infrastructures and businesses alike are vulnerable to DoS attacks or flash-crowds that can incapacitate their networks with traffic floods. Unfortunately, current mechanisms require per-flow state at routers, ISP collaboration, or the deployment of an overlay infrastructure to defend against these events.

In this paper, we present SIFF, a Stateless Internet Flow Filter, which allows an end-host to selectively stop individual flows from reaching its network, without any of the common assumptions listed above. We divide all network traffic into two classes, privileged (prioritized packets subject to recipient control) and unprivileged (legacy traffic). Privileged channels are established through a capability exchange handshake. Capabilities are dynamic and verified statelessly by the routers in the network, and can be revoked by quenching update messages to an offending host. SIFF is transparent to legacy clients and servers, but only updated hosts will enjoy the benefits of it.

1 Introduction

Despite a significant breadth of research into defenses, Denial of Service (DoS) attacks remain a significant problem in the Internet today. The DoS phenomenon has evolved rapidly over the last decade. DoS attacks were once caused by only a few attackers—often only a single attacker—sending specially crafted packets designed to exploit flaws in the victim's particular TCP/IP implementation (e.g., the Teardrop or Land Attack), and sometimes using IP spoofing [10] (the forging of the source IP address field in the IP header to something other than the sending host's IP address) to hide their identity. DoS attacks are becoming an increasing risk, as the sophistication of current attack tools enables relatively inexperienced attackers to perform these attacks.

As the number of systems connected to the Internet has increased, the black-hat community has developed tools (known as *root-kits*) that take advantage of security flaws in operating system services to compromise computers. The black-hat community has also written tools designed to control and coordinate these exploited machines (often called *zombies*) on a large scale [13, 14]. These developments have given rise to a new type of DoS classification: the distributed DoS attack (DDoS). These attacks tend to be different from simple DoS attacks in that the attacks are composed of compromised hosts that are not easily traceable to the machine controllers themselves. For this reason, DDoS attackers are not concerned with spoofing as a disguising tactic; they merely use it to bypass potential IP address filters or to involve unwitting third parties in the attack as traffic amplifiers [34]. Furthermore, because of the sheer number of hosts involved, (there have been reports of groups claiming over 140,000 compromised hosts under their control [11]), the attacks tend to work simply by flooding packets onto the network, which converge upstream from the intended victim and disrupt the infrastructure, to prevent or reduce legitimate access to the victim.

The DDoS flooding problem is particularly difficult to defend against, because the very architecture that has fueled the Internet's growth—reliance on intelligent end-hosts connected by a relatively simple network (the end-to-end principle)—is used to the attacker's advantage. In this network architecture, any host in the network can send a packet

*This research was supported in part by the Center for Computer and Communications Security at Carnegie Mellon under grant DAAD19-02-1-0389 from the Army Research Office, and by gifts from Bosch, Cisco, Intel, and Matsushita Electric Works Ltd. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of ARO, Bosch, Carnegie Mellon University, Cisco, Intel, Matsushita Electric Works Ltd., or the U.S. Government or any of its agencies.

to any destination address (even if the destination does not want to receive the packet), and the destination has no way of stopping those packets before they reach it (or its network). Although many innovative avenues of defense against DoS and DDoS have been explored in the literature (we review some of these in the next section), only a few of the approaches even address the DDoS flooding problem. In this paper, we explore the design issues involved in constructing a system from scratch that solves the DDoS flooding problem by giving a packet receiver control over which packets the network delivers to it.

1.1 Related Work

Several researchers have studied the frequency and nature of Internet DoS attacks [16, 17, 21, 24, 31]. In this section, we review related work in the area of Internet DoS defenses. We distinguish the research into two general areas: defending against IP source address spoofing, and defending against bandwidth flooding attacks.

We first discuss research in defenses against source IP address spoofing. Ferguson and Senie propose to deploy network ingress filtering to limit spoofing of the source IP address [15]. Burch and Cheswick propose to use a limited form of DoS attack to probe which links are affected by an attack and can thus trace back to the origin [8]. Park and Lee propose a distributed packet filtering (DPF) mechanism against IP address spoofing [32]. DPF relies on BGP routing information to detect spoofed IP addresses.

Bellovin et al. suggests adding a new type of ICMP message for traceback [6], and Mankin et al. present an improvement to this scheme [28]. Several researchers propose to embed traceback information within the IP packet [2, 12, 19, 26, 35, 38, 43]. Most of these schemes use the 16-bit IP Identification field to hold traceback information. Routers along the packet's path probabilistically mark bits in the IP Identification field in different ways. While traceback schemes could be used to find the origins of the attacks, they often require a large number of packets and cannot be used to filter out packets on a per-packet basis.

Snoeren et al. propose a mechanism using router state to track the path of a single packet [37]. Their approach enables a victim to trace back a single packet by querying the router state of upstream routers. In earlier work, we propose the Pi marking scheme to enable the victim to detect packets with a spoofed source IP address [47]. Pi is not effective against bandwidth flooding attacks because it relies on victim filtering of DDoS traffic and bandwidth flooding typically causes damage (i.e., dropped packets) upstream from the victim.

The IP traceback and spoofing defenses we discussed so far, defend against DDoS at the victim of the attack. The research most closely related with our work attempts to defend against network flooding attacks in the network itself. Stone proposes the CenterTrack mechanism, which uses routers capable of input debugging (the ability to determine through which router interface a particular packet was received) that would be virtually connected through IP tunnels to all border routers on a network [42]. 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 determine from which border router, and hence from which neighbor network, the DDoS traffic is coming from. However, this technique requires that the ISP create the overlay network and perform filtering, and may not be practical for large attacks.

Generalized network congestion control is related to DDoS defense. Ioannidis et al. propose Aggregate Congestion Control (ACC)/Pushback, which leverages router support to rate-limit groups of similar packets responsible for congestion, and push filters upstream towards the sources of those packets to preserve network bandwidth [22, 27]. ACC/Pushback requires non-negligible state at routers and faces challenges in attack traffic identification and ISP interoperation. Stoica et al. propose a mechanism for Stateless Fair Queueing (SFQ) in the network core [40]. Their scheme has edge routers maintain per-flow arrival-rate information. Edge routers label packets based on their flow's rate information, and core network routers use probabilistic dropping based on the packet labels to fairly distribute bandwidth among flows. SFQ requires that the malicious flows' edge routers implement the labeling, so that those flows can be rate limited in the core.

Researchers recently investigated using overlay networks to mitigate the effect of DoS attacks. Keromytis et al. designed the Secure Overlay Services (SOS) architecture to proactively defend against DoS attacks [25]. SOS uses an overlay network to authenticate users and installs filters at the ISP level. Anderson generalizes the SOS architecture by considering different filtering techniques and overlay routing mechanisms and proposes Mayday [4]. Adkins et al. propose the use of the Internet Indirection Infrastructure (*i*3) [39] to enable the victim to stop individual flows by removing the unique forwarding pointer that each sender possesses [1]. An advantage of these techniques is that they do not require changing the current Internet infrastructure. However, they assume presence of an overlay infrastructure, they require per-flow state in the overlay network, and they assume updated client and server software and protocols.

Mirkovic et al. propose D-Ward, an automated system that detects flooding attacks on a link and automatically installs filters [29]. In their system, the receiver has no control over these filters, and so the receiver has no way of stopping a widely distributed flooding attack or choosing flows it wants to serve in case of a flash crowd [24]. Jamjoom and Shin propose persistent dropping for dealing with flash crowds [23]. Their drop policy preferentially drops TCP SYN packets in favor of preserving ongoing TCP flows, but does not deal with DoS attacks that may flood with traffic other than TCP SYNs.

Our SIFF protocol or a capability-based approach that uses per-flow state can easily be implemented in a secure active network environment [3].

Anderson et al. present a capability-based approach, where a sender first needs to obtain a capability from the infrastructure to send traffic to a receiver [5]. Their approach is similar to ours, but requires per-flow state at each router.

Gligor analyzes various mechanisms for filtering excessive connection establishment requests [18]. Assuming that the network connections are not saturated, he analyzes the waiting time guarantees that various puzzle and client challenge techniques provide and comes to the intriguing conclusion that puzzles are not useful to provide any guarantees of waiting time.

In contrast to these previous approaches, the mechanisms we present in this paper provide the victim of a flooding attack (or a server in the case of flash crowds) to select individual traffic flows that it wants to stop, without requiring per-flow state in the network (in fact, without contacting any routers or ISPs), while still enabling legacy clients or servers to communicate with updated servers or clients.

1.2 Organization

The remainder of our paper is composed as follows: in Section 2 we define the properties and assumptions that we make in designing SIFF. In Section 3 we present a detailed design of the SIFF system and protocols. In Section 4 we present a model, based on real Internet topologies, to analyze SIFF's performance. In Section 5 we discuss several possible modifications and extensions to SIFF. Finally, in Section 6 we conclude the paper.

2 Problem Statement and Assumptions

Several researchers recently report that while the victim is in the best position to detect flooding DDoS attacks, it lacks the means to stop malicious flows in the current Internet architecture [1, 5, 18]. Similarly, flash crowds exhaust the bandwidth on network links and TCP flows mutually prevent each other from achieving high bandwidth [24] (the additive increase when there is no packet loss and multiplicative decrease when there is packet loss causes TCP connections to reduce their bandwidth to a very small amount if packet loss exceeds 5%). Thus, there is the need for a mechanism that enables the victim of a flooding attack or the server of a flash crowd to be able to stop individual flows before they saturate its network, to provide good performance to the remaining flows. In this paper, we present SIFF, a Stateless Internet Flow Filter that enables an end-host to stop individual traffic flows from reaching it, without keeping per-flow state in the network. Specifically, SIFF provides the following properties:

- **Client/Server privileged communication.** SIFF allows a client and server to establish a privileged channel over IP whose packets take precedence over non-privileged packets.
- **Recipient controlled privileged flows.** SIFF gives the receiving host of a privileged communication channel the ability to tear-down that channel, and thus stop the flow of packets on the channel from reaching its network. Packets from that channel will get dropped by a router close to the sender with, high probability, and thus these packets will not take up bandwidth resources on a link close to the receiver.
- **Limited spoofing of source addresses.** Equivalent to ingress filtering—the receiving host of a privileged communication channel can be sure, with high probability, that the packets arriving on that channel were sent by a host on the same network as the host having the source IP address in the packet.
- **No end-host/ISP or inter-ISP cooperation.** SIFF does not require end-host signaling of routers or the signaling of routers of one ISP by those of another ISP.
- **No intra-ISP cooperation.** SIFF does not require signaling between a single ISP's routers beyond that required for packet routing.
- **Small, constant state at routers.** Routers implementing SIFF need only keep a constant amount of state per router interface, independent of the number of privileged channels traversing that router. This is one of the main features of SIFF, as other mechanisms we are aware of require per-flow state at routers to achieve the above properties.
- **Small, per-packet processing at routers.** A SIFF router need only execute two equality checks for every privileged packet, or a single hash computation (which can be reduced to a table lookup) for every unprivileged packet, that it forwards. These computations are independent of the number of privileged or unprivileged channels traversing a router. The hashing and equality checks can be done in parallel with the routing table lookup, though in practice the equality checks can serve as a packet filter to prevent extraneous lookups.

- **Backward Compatibility.** Legacy clients and servers do not break SIFF, and legacy clients can communicate with updated servers and vice versa. However, both clients and servers must be updated to take advantage of the system's benefits.

To construct a system with these properties, we begin with the following assumptions, some of which we use for simplicity of presentation and will relax or remove in Section 5.2.

We first assume that a victim has the ability to determine that it is under attack, and can differentiate between legitimate client flows and malicious or misbehaving flows. We do not require that this differentiation be on a per-packet basis, or that it be lightweight; only that it exist. [1] However, the details of a traffic differentiation algorithm are application specific and orthogonal to the focus of this paper, which simply assumes their existence.

Secondly, we assume that clients, servers and routers are redesigned and conform to a modified IP network layer (non-updated clients and servers will still be able to communicate with updated clients and servers, but they will not realize the benefits of the new system). Specifically:

- **Marking space in the IP header.** We assume that the IP header has sufficient space to accommodate the information that routers mark in the packet.

- **Routers mark every packet.** We assume that SIFF routers are capable of executing minor manipulations of the marking field of every packet that they forward. These manipulations can be done in parallel with a routing table lookup. This assumption is minor, since Internet routers must already decrement the TTL and recalculate the IP Header Checksum of every packet they forward.

- **Short-term Route Stability.** We assume that Internet routes are stable on the order of the time of a client/server transaction. Violation of this assumption will not break our system outright, rather, the system's performance is likely to decrease with increasing route instability below the time required for a client/server transaction. Network routes are more likely to fluctuate under DDoS attack, precisely when our system requires their stability. However, SIFF will also mitigate the effect of DDoS on routers (as packet floods are dropped early in the network), and is, in this way, self-reinforcing. Unfortunately, it is difficult to model the behavior of a system as complex as the Internet, especially under DDoS attack, so verification of this assumption is an open problem.

[1] Because our mechanism limits source address spoofing, it can make malicious host identification easier.

Our approach divides all Internet traffic into two types, privileged and unprivileged. Privileged packets are *always* given priority over non-privileged ones when contending for bandwidth. To establish a privileged channel, a client must obtain a capability through a special handshake over an unprivileged channel. Privileged channels consist of specially marked packets embedded with the capability obtained through the unprivileged handshake.

The capabilities in SIFF are based on information inserted into all packets by the network en route to their destinations. This mechanism is similar to that of Pi, proposed by us in an earlier paper [47], except that in SIFF, the computation process for markings is slightly more elaborate, as packet markings change over time (as opposed to remaining constant in Pi), and are used by both routers and endhosts, rather than just by endhosts. The capability is generated piecemeal by each router and marked in a field of the packet along the path to the packet's destination. The pieces at each router are generated entirely from packet header data and local topology information.

SIFF provides the above benefits to the receiver of a privileged channel. When forwarding a privileged packet, a router simply checks part of the embedded capability to see if it matches the markings that the router would have added to an unprivileged packet—if they match, then the packet is forwarded; if they do not, then the packet is dropped. The capabilities themselves are based on the packet markings, which change independently at each router with a certain frequency. Routers maintain a window of valid markings and signal a change of marking to a packet recipient by replacing old markings in the embedded capability with new ones. Because the packet recipient, rather than the packet's sender, is receiving the capability updates, continued privileged communication requires that the receiver periodically update the sender's capability. Thus, the receiver of a packet flow has the option to halt that flow by simply refusing to forward capability updates. Attackers can still flood using unprivileged packets, but they will no longer disrupt existing privileged communications. Furthermore, during an unprivileged flooding attack, a legitimate client and server need only pass two packets (a total of one round trip) between themselves to establish a privileged channel and communicate, undisturbed, over privileged packets.

3 Design

The SIFF system provides a server with the ability to establish privileged communication with whatever clients it chooses by providing those clients with a capability token. [2]

[2] For ease of presentation, we refer to a flow's source as *the client* and a flow's destination as *the server*. This does not mean that privileged channels can only be established from clients to servers.

Privileged packets carry capabilities that are verified piecemeal (and statelessly) by the routers in the network, and are dropped when the verification fails. Routers implementing SIFF are programmed to give preferential treatment to privileged packets, so that privileged packets are never dropped in favor of unprivileged ones (legacy packets not conforming to our scheme are treated as unprivileged packets). Privileged channel capabilities are time limited and require updating by the server to remain valid. Because server cooperation is required for capability updates, a server can halt the packets of a privileged channel by simply quenching its capability update messages.

At a high level, the system works as follows: clients and servers participate in a handshake (similar to the TCP handshake, which can be carried on top of this handshake) using a specific type of unprivileged packet known as an *EXPLORER* (or EXP) packet. Routers insert path specific information into EXP packets, who's aggregate among all the routers in the path is used as a capability token for a privileged channel between the client and the server. After the handshake, clients and servers communicate using privileged packets called *DATA* (or DTA) packets, into which they insert the capabilities carried in the EXP packets. When routers forward a DTA packet, they first check to see if part of its capability equals that information which would have been inserted into the packet had it been an EXP packet. If the markings match, then the packet is forwarded. If not, then the packet is immediately dropped.

Our discussion assumes a new format for the IP header. The following fields are assumed to be present:

- **Flags field** (3-bits). This field contains the following 1-bit flags: the signalling flag (`SF`), used to indicate that the packet is a non-legacy (either EXP or DTA) packet; the packet type flag (`PT`), used to indicate that the packet is either a DTA (set) or EXP (unset) packet; and the capability update (`CU`) flag, set to indicate that the optional capability reply field is present in the header.

- **Capability field**. This field is used by routers to add their marks to the packet en route to its destination.

- **(Optional) Capability reply field**. This field is used by packet recipients to signal to the packet sender a new (or updated) capability, and is only present when the capability update flag is set.

We do not assume an exact length for the capability or capability reply fields, as their lengths will depend upon other parameters (such as the bits marked per router and maximum path length). We assume the presence of a source and destination address in the header, but not their exact length. No other fields of the packet header are used in our scheme.

In the following subsections, we describe in detail the handshake protocol, as well as the potential issues in its implementation.

3.1 Handshake Protocol

Any client wishing to contact a server over a privileged channel must first complete a handshake protocol to obtain a capability to insert into its privileged packets, and vice versa for server communication with the client. A single handshake is sufficient to provide both sides of a communication with their capabilities. Furthermore, handshake packets can carry upper layer protocol data. The protocol is shown in Figure 1.

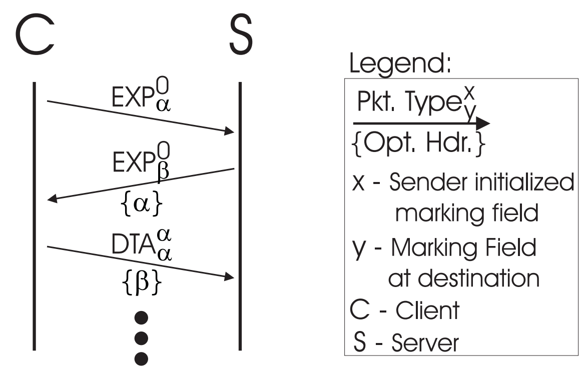
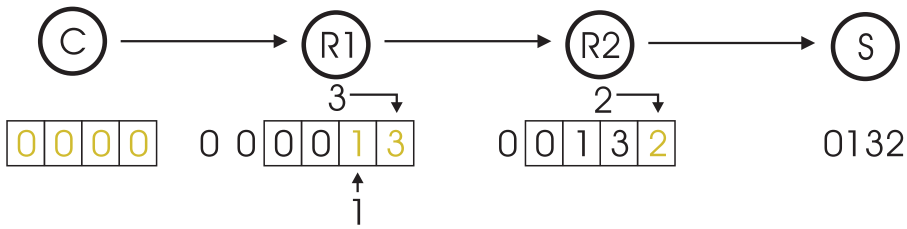
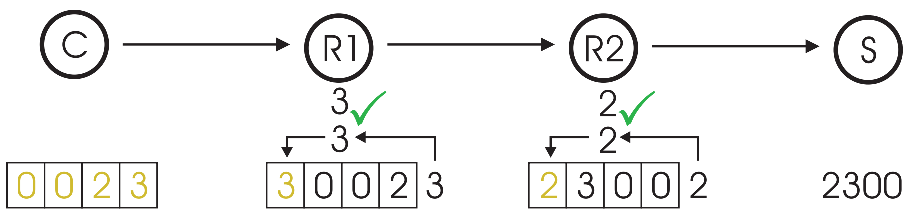


Figure 1. Handshake establishing a privileged channel. A client sends an EXPLORER packet to the server, which gets marked with marking α. The server responds with its own EXPLORER packet, with α enclosed in the capability reply field. The client sends its first DATA packet with α in its capability field and with β, from the server's EXPLORER packet, enclosed in the capability reply field.

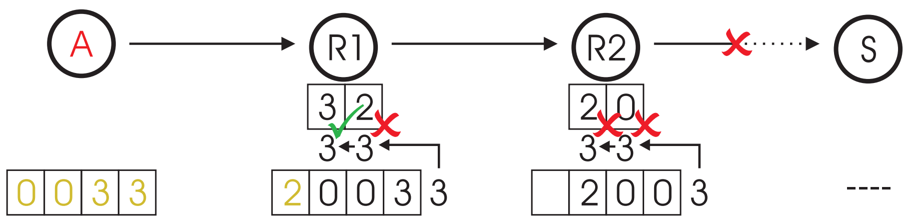
The initiator of the handshake (the client) first sends out an EXP packet with its `Capability` field initialized to 0. A packet is marked as an EXP packet by setting the signalling (`SF`) flag and leaving the packet type (`PT`) flag unset. All routers along the path left shift z bits into the `Capability` field the EXP packet (see Section 3.2 for a description of how these markings are computed). The exception to this rule is that the first router in the path that sees a marking field of all 0 bits inserts a 1 bit before its marking (so that the actual capability in the field will consist of all bits up to, but not including, the most significant 1 bit.). Recall from Section 2 that we assume that the marking field is large enough to accommodate the markings from all of the routers in the path plus the 1 bit inserted by the first router. EXP packet marking is shown in Figure 2(a).



(a) Marking scheme for EXPLORER packets. Routers push their markings into the least significant bits of the capability field. Packets with a capability field of all zeros get marked with an additional 1 bit.



(b) Authentication scheme for DATA packets. Routers check the marking in the least significant bits of the capability field, and rotate it into the most significant bits, if it is equal to what the marking would be for an EXPLORER packet.



(c) Windowed authentication and marking for DATA packets. Routers check that the marking equals one of the valid markings in its window and always rotate the newest marking in the window into the capability field.

Figure 2. Marking and authentication schemes for EXPLORER and DATA packets.

When the EXP packet arrives at the server, the server creates a response packet. The response packet is also an EXP packet, with the `Capability` field initialized to zero, but with the capability update (`CU`) flag set, and the `Capability Reply` field initialized to the contents of the `Capability` field of the EXP packet from the client. When the server's EXP packet arrives at the client, the client examines the `Capability Reply` field, takes all the bits up to— but not including— the most significant 1 bit in the packet, splits them into groups of z bits and reverses the order of the groups to obtain its capability. This capability is inserted into the `Capability` field of all subsequent privileged packets that the client sends. To complete the handshake, the client must send the server its capability, marked by the routers in the server's EXP packet's `Capability` field. The client creates a DTA packet, with the `CU` flag set and the `Capability` field from the server's EXP packet in the `Capability Reply` field; just as the server did for the client.

The router behavior for marking and forwarding DTA packets is different from that used for EXP packets. When a router receives a DTA packet, it calculates a marking as though the packet were an EXP packet, but then only verifies that the marking it has calculated is equal to the marking in the least significant bits of the `Capability` field. If the marking is not equal, then the packet is immediately dropped. If the marking is equal, then the router right shifts that marking into the most significant bits of the `Capability` field. This causes the markings for the next hop router to occupy the least significant bits. DTA packet marking is shown in Figure 2(b). A DTA packet will only reach its destination if all the routers along its path match their markings to the least significant bits of the packet's `Capability` field. Once the client's privileged DTA packet arrives at the server, the server can compute its capability in the same way that the client did and the handshake is complete, as both hosts can now communicate over privileged DTA packets.

3.2 Router Marking Calculation

As described in the previous section, each router must calculate a marking for every packet that it forwards; it left-shifts the marking into the packet in the case of an EXP packet, or verifies and right-shifts the marking in the case of a DTA packet. For a particular packet, the marking is calculated as the last z bits of the output of a keyed hash function with the following parameters as input: the IP address of the interface at which the packet arrived at the current router, the last-hop router's outgoing interface IP address [3], and the source and destination IP addresses of the packet being forwarded.

The use of the source IP address as a hash input has the effect of tying a capability to a particular host and eliminates the effect of source address spoofing. If the attacker is on a shared medium network with a legitimate client and observes a capability transmitted to that client, the attacker is limited to spoofing the client's IP address when flooding using that client's capability. The server will revoke the capability (using a mechanism we introduce in Section 3.2.1) and all packets using the client's capability will be dropped from the network. Although this results in a DoS on the client, the attacker can presumably accomplish the same goal by simply ignoring the transmission control mechanism of the shared medium it occupies.

The use of the destination IP address as a hash input prevents attackers from generating "marking maps" by sending EXP packets from one attacker to another and observing the marks that result. Any marks learned in this fashion will be invalid when used to flood DTA packets to a different machine. [4]

For SIFF to effectively stop forged privileged packet floods, a router must be able to calculate its marking faster than it can perform a route lookup. Otherwise, attackers could simply flood a router with illegitimate DTA packets, causing the router to either overload its route-lookup capability or fill its buffer with DTA packets, and start dropping potentially legitimate DTA packets indiscriminately. To meet this performance goal, the router must be able to calculate the hash function in hardware. Alternatively, the router could select a random subset of bits of the source and destination IP addresses to use in its hash function (it is assumed that IP_{i-1} and IP_i change infrequently), and precalculate the hash results using all possible permutations of these input bits. Of course, the bits of the addresses used for the hash calculation would have to change periodically to avoid attacker's identification of them, perhaps as part of the mechanism described in Section 3.2.1.

EXP—or legacy—packet floods do not interfere with DTA packets, because of the priority given to DTA traffic. However, even under these floods of packets the overall performance of the router need not suffer, because the marking calculation (or precalculated marking lookup table) can be executed in parallel with the routing lookup in the router because the marking does not depend on any routing decisions.

3.2.1 Key Switching

Thus far, the router information inserted into packets has been treated as static and unchanging. If this were the case, than an attacker could simply obtain a capability through a seemingly legitimate request, and then use that capability to flood the server with privileged traffic. To meet the design goal of allowing a server to stop a malicious flow that is already in progress, we introduce the router key switching mechanism.

To prevent an attacker from abusing a legitimately acquired capability, we require that the capabilities in our system change over time. This is accomplished by having the routers change their markings periodically, by changing the keys to the hash functions they use to compute the markings. However, rather than invalidate an entire capability and force the client and server to execute another handshake simply because one router changes its key, routers keep a window of x keys as valid at any one time, where $x > 1$. When a privileged packet with an old marking (i.e. one still present in the router's window, but not the most recently computed one) arrives at the router, the router shifts the most recent marking into the packet, thus providing the server with an updated capability. The server can then signal its client of the updated capability using the `CU` and `Capability Reply` mechanism, if the client is well-behaved. [5] Figure 2(c) shows window verification and

[3] The last-hop router's IP address is used to improve the entropy of the marking [47].

[4] A more serious vulnerability for a server would be to have a colluding or unwitting host on its network respond to EXP packets, thus providing an attacker with a capability that can be used to flood privileged DTA packets to the server's network. However, this problem can be avoided by careful administration of the victim's network.

[5] A server in our scheme can be implemented with or without state. A stateless receiver does not need to keep track of the capability for each client and would simply reply to any packet by setting the `CU` flag and filling in the `Capability Reply` field with the value in the latest privileged packet's `Capability` field from that client. If the client is deemed

marking of privileged packets.

4 Analysis

4.1 Privileged Packet Flooding

SIFF mitigates the impact of flooding (or bandwidth starvation) DoS attacks by isolating and protecting established privileged communication from unprivileged communication and enabling the receiver to downgrade privilege. In this section, we analyze the robustness of our scheme against floods of privileged packets with forged capabilities.

First, we derive the probability that an unauthorized router or end-host will be able to forge (by guessing) the appropriate capability to allow its packets to reach a potential victim. Recall from Section 3 the two parameters: x, the number of markings each router maintains in its window; and z, the number of bits per router marking. The probability that a randomly guessed capability will pass a particular router is given by:

$$P(x,z) = 1 - \left(1 - \frac{1}{2^z}\right)^x$$

and the probability that a randomly guessed capability will pass all d routers in a path is simply $P(x,z)^d$.

Recall from Section 3 that the capability field *must* be large enough to accommodate the markings of all routers in the path (plus an additional 1 bit to indicate the first marking in the field), or else the markings of some routers will be pushed out of the field, and the capability (when used by the sender of that packet) will not pass the inspection of those routers whose markings were dropped. Table 1 shows $P(x,z)$, the probability of successfully forging a capability to pass a single router, for reasonable values of x and z.

	x = 2	x = 3	x = 4	x = 5
z = 1	0.7500	0.8750	0.9375	0.9688
z = 2	0.4375	0.5781	0.6836	0.7627
z = 3	0.2344	0.3301	0.4138	0.4871
z = 4	0.1211	0.1760	0.2275	0.2758

Table 1. Evaluation of $P(x,z)$ (the probability to pass one router with a forged probability), for common values of x and z.

Choosing a value for x (the maximum number of keys that are valid at any one time on a router) presents a trade-off. As we show in Section 3.2.1, x must be at least 2, otherwise every key change would force an additional handshake between the client and server. Because a valid capability is one that matches any of the x capabilities in the router's window, small values of x provide the smallest probability that a randomly chosen capability will pass through a router. (Table 1 shows this effect). The value of x also affects the validity time of a capability. The minimum validity time (min_m) is $min_m = (x-1) \cdot T_K$, where T_K denotes the time a key (marking) is active (valid). The maximum validity time (max_m) is $max_m = x \cdot T_K$. Ideally, we would like to get a small interval for the validity time, so that we can tightly control the validity period, so we would like large values of x to minimize the difference between min_m and max_m. We can determine x from min_m and max_m:

$$x = \frac{max_m}{max_m - min_m}$$

Because capabilities are only updated when the client and server send packets to each other, the min_m metric should be set just low enough so that all handshakes and most packet traffic should be able to go back and forth from client to server within that time period. The max_m metric defines the longest amount of time that a connection can remain idle and still have a valid capability. Put in another way, max_m defines the maximum amount of time that an attacker can flood privileged packets with a particular capability, before that capability is rejected by the network. We leave the exact timing decisions to the community, and simply assume in our experiments that x is likely to be from two to five.

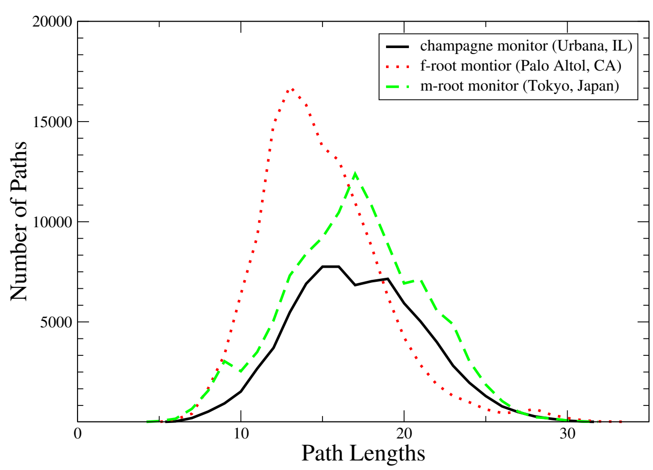
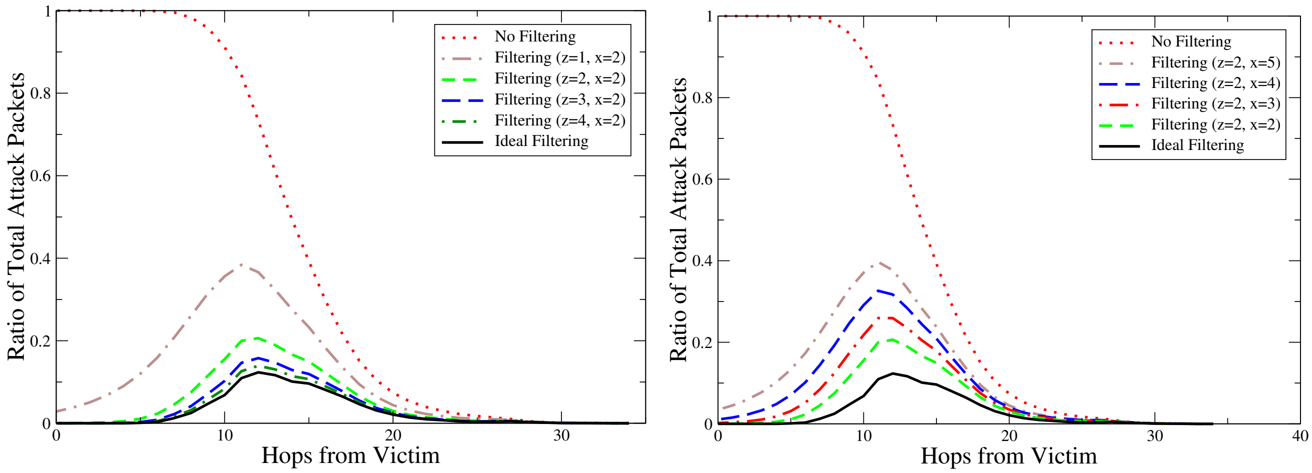


Figure 3. Path length distribution for three CAIDA skitter monitors.

In Figure 3 we show the plots of the path lengths of the Internet maps generated at three different CAIDA skitter monitors [9]. To analyze the performance of our scheme in filtering floods of forged privileged packets, we select

malicious, the automatic reply can be stopped. Although always responding with a capability update wastes bandwidth, we present this mechanism for those concerned with maintaining IP as stateless at the endhosts. A stateful implementation of this system is straightforward, and preferred.



(a) Performance for various values of z, (x = 2).

(b) Performance for various values of x, (z = 3).

Figure 4. Packet filtering performance for varying bits per router (z) and window sizes (x).

attackers at random from our map, and have them send a number of packets with randomly forged capabilities. The number of attackers and the number of packets per attacker do not affect the outcome of our experiment, in which we simply drop a certain percentage of attack packets at each hop. We show the results of our experiments only from the f-root skitter monitor; which are the most pessimistic because of the high concentration of paths close to the victim relative to the other two monitors' path distributions.

Figure 4(a) shows the ratio of total attack traffic at each hop from the victim for varying values of z. As expected, without filtering of any kind, eventually all attack packets arrive at the victim. Furthermore, with ideal filtering (routers automatically drop all attack packets) we see a curve that matches the path distribution, since the attackers' packets are immediately dropped after one hop. The SIFF scheme performs excellently, filtering out 97.14% of the attack traffic using only a one bit marking per router (z=1), and filtering out 100% of the attack traffic (six nines) with a marking scheme of four bits per router (z=4).

Figure 4(b) shows the same experiment with a constant z and a varying x. As expected (and suggested by Table 1) the effect on filtering performance caused by varying z is far greater than that caused by varying x. Furthermore, although not shown in the figure, it is intuitive that the effect of varying x decreases as z increases.

4.2 Privileged Channel Establishment Under Unprivileged Packet Flooding

As shown in the previous section, attacker flooding of privileged packets has little effect on the victim, because so few of the forged packets reach destinations even close to the victim's network. In this section, we analyze a different attack approach, which is to flood with unprivileged packets for an extended period of time with the goal of stopping all new connection establishments. However, unlike the current Internet infrastructure, in which established TCP flows can still be affected by IP packet floods, SIFF's privileged flows are unaffected by unprivileged traffic congestion. Thus, a client and server only need to exchange two packets within min_m time, the least amount of time that a capability is valid (defined in the previous section), before the privileged channel between them is established and they can communicate from then on, unaffected by the ongoing attack.

We assume that unprivileged traffic is causing congestion at the last i hops of the network, and that the probability of getting dropped at any one of those routers is ϵ_i. We ignore the probability of the server getting its outbound packets dropped, because congestion in the routers during flooding attacks is typically experienced by inbound packets only. Because the drop probabilities at routers are independent Bernoulli trials, the probability that a client and server will be able to establish a privileged channel after one try (by exchanging two packets is): $P(\text{connect after 1 try}) = (1-\epsilon_i)^i$.

The probability that the client can connect after k tries is:

$$\begin{aligned} &P(\text{ connect after k tries}) \\ &\quad = 1-(1-P(\text{connect after 1 try}))^k \\ &\quad = 1-(1-(1-\epsilon_i)^i)^k \end{aligned}$$

For a given desired connection probability, $P(connect)$

the required number of connection attempts is:

$$k = \frac{\log(1 - P(connect))}{\log(1 - (1 - \epsilon_i)^i)}$$

A nice feature of this formula is that the expected number of connection attempts depends logarithmically on the connection probability, which indicates that even for large ϵ_i, a determined client can get a connection after a moderate waiting time.

5 Discussion

In this section, we discuss some limitations, practical issues and extensions to SIFF. We first discuss several classes of bandwidth starvation attacks against which SIFF does not completely defend. We also discuss a high-level approach for implementing SIFF in an IPv4 environment, the combination of SIFF with puzzle auctions, the possibility of multiple capabilities with different validity lengths, and the effect of route stability on our scheme.

5.1 Limitations

We have identified several types of bandwidth attacks that SIFF does not defend well against. We briefly describe them in this section.

Although SIFF binds a source IP address to a particular capability (thus limiting the possibility of spoofing in privileged flows to the same as ubiquitous ingress filtering), without a mechanism to identify malicious traffic, it is still possible for an attacker to rotate the active machines in its attack. Presumably, when a subset of attacking machines are blacklisted the attacker would activate a different subset to request (and abuse) new capabilities. However, depending on the size of the victim's blacklist, the attacker will be limited in the number of active machines used at any one time in the attack.

In a topology where not all routers implement SIFF, it may be possible for a carefully placed bandwidth attack to disrupt privileged communication. If the attacker pinpoints a link where a router does not implement SIFF, then by flooding that link with unprivileged traffic he can cause privileged traffic to be dropped, because the router at that link will not give priority to privileged traffic. To prevent this attack, it is sufficient that the router on the attacked link implement the preferential treatment of privileged packets, rather than the whole SIFF protocol.

The case of colluding attackers is difficult to solve in SIFF. As mentioned in Section 3.2, a colluding attacker node on or near the victim's network can return capabilities to attacking nodes outside the network. In general, colluding attackers on either side of a transit network may simply grant each other capabilities and flood the network with privileged traffic, causing other privileged traffic traversing that network to experience congested links. A possible solution to this attack would be to have routers rate-limit individual capabilities, similar to the Aggregate Congestion Control/Pushback mechanism [27].

It is also important to note that, as designed in this paper, SIFF is a layer-3 (IP) mechanism. As such, SIFF can grant capabilities only at the granularity of hosts (i.e., single IP addresses). For example, a SSH server cannot give privilege to an SSH client without also granting privilege to any other connection from that host.

5.2 Deployment in the Current Internet

The limited space in the IPv4 packet header and the limited deployment that any routing infrastructure change is likely to achieve makes it necessary to redesign SIFF. Fortunately, both of these constraints can be satisfied by a single approach: rather than constructing a system where forged privileged flows are dropped anywhere in the network we focus on hardening individual ISP's against such traffic flows. We assume two possible deployment models: *full ISP deployment* and *border ISP deployment*. In the full ISP deployment model, all routers under the control of a particular ISP are upgraded with our scheme, whereas border ISP deployment requires only that all of an ISP's border routers be upgraded. The intuition behind both approaches is that by limiting the number of marking routers (to just the routers of the packet's destination), each router can mark more bits in the available marking space and forged privileged flows can still be stopped, albeit only once they arrive at the victim's ISP's domain. Note that end-hosts, both clients and servers, will still need to be modified to take advantage of these schemes.

SIFF requires some available space for marking in the IPv4 packet header. We require one bit of the IP header that is currently reserved (set to zero by all end-hosts) to act as the signalling flag (`SF`) which, as mentioned in Section 3, will differentiate legacy traffic from all traffic used in our scheme. We do not assume any particular location for the remaining markings, although of course, any location chosen should avoid interaction with legacy protocols (we offer some insight into how to avoid interaction with fragmentation when marking the IP Identification field in Appendix A). Furthermore, we remove the capability update (`CU`) flag and assume that half of our available marking space is used for capability replies in every packet. The packet type (`PT`) flag remains unchanged. Thus, if we assume x bits available in the IPv4 packet, then $\frac{x-2}{2}$ bits are used for capability marking, $\frac{x-2}{2}$ bits are used for capability replies, and 2 bits are used for flags. We show SIFF's performance for varying numbers of marking bits in Figure 5. The figure shows a significant reduction in the percent of total attack traffic arriving at the victim, ranging from 25% to 98.7%, depending on the number of marking bits used.

Finally, we address a security hole in the border ISP deployment method. An attacker could determine its capability by simply sending a packet designed to produce an ICMP error message at a router between the victim and the ISPs border routers (for example, a TTL expiration). The ICMP error packet sent by the router will include in its payload the IP header and the first 8 bytes of the payload of the original packet, thus returning to the attacker the capability that will bypass the ISPs border routers. An approach to prevent this attack is to have all the border routers of the ISP's network monitor outbound ICMP error messages and remove the contents of the marking field in messages that contain EXP packets. Although this may degrade performance on border routers, ICMP has a simple header, so packet inspection can be implemented in hardware. ICMP attacks are not a problem for full-ISP deployment because capability enabled routers can be programmed to mask out the marking field of all EXP packets before they are encapsulated in ICMP error packets. Border ISP deployment is also subject to the legacy router bandwidth attack described in Section 5.1.

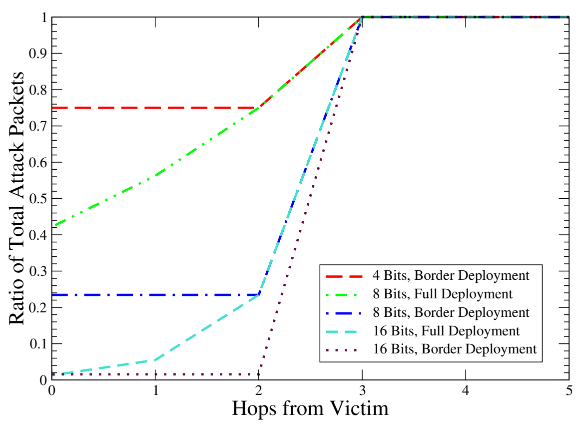


Figure 5. Percentage of attack packets in the last 5 hops to the victim for different marking sizes and deployment methods.

5.3 Puzzle Auctions

SIFF can be combined with Wang and Reiter's puzzle auctions [46], to minimize the assumption that a server needs to differentiate between legitimate and malicious clients. The intuition behind puzzle auctions is that a client makes a bid as to the difficulty of the puzzle it is willing to solve in order to receive a capability. Presumably, compromised machines are unwilling (due to the chance that their users would discover the compromise) or unable (due to the high frequency of their capability requests) to devote as many resources to solving a puzzle as a legitimate client.

In a combined scheme, a client would transmit a value x in its initial EXP packet representing the difficulty of the puzzle it wishes to solve. The server will receive the capability y of l bits in length and respond with $y_{l-(x+1)}|h(y)_{x-0}$ (where $h(y)_{x-0}$ is the least significant x bits of the hash of the capability y, and $|$ is concatenation). The client must perform 2^{x-1} hash operations, on average, before finding the correct pre-image of the last x bits of the capability. This scheme could be used without adding any fields beyond those already assumed in Section 3.

Puzzle auctions, in this context, have the disadvantage that they reduce the search space for an attacker trying to forge valid capabilities. Furthermore, there is a limit to the difficulty of the puzzle given to the client, because the capability contained within the puzzle may expire while the client is solving it.

5.4 Path Stability Effects

In Section 2 we assume Internet route stability (on the order of a client transaction). If a route changes mid-flow, then a client's privileged packets will be dropped by the new routers in the path (with high probability), and it will force the client to renegotiate the SIFF handshake before being able to send further privileged packets.

Route instability can be caused by multiple path effects. Teixeira et al. analyze the CAIDA skitter topologies to show that at least half of the endpoints in the Internet have more than 2 partially-disjoint paths (where there are no common routers *between* the source and destination ASes) [44]. However, this result is orthogonal to our system's performance as long as routing decisions between multiple equal-cost paths are implemented in a flow preserving way (eg. by hashing the flow fields of the TCP/IP packet headers) as suggested by RFC2992 [20]. Furthermore, large-scale route flapping has been shown to be detrimental to TCP performance due to the difficulty in estimating path characteristics such as round trip time [33]. Localized load-balancing does not hurt TCP performance, but we assume that local load balancing nodes can be manually configured to produce the same markings.

An alternative SIFF forwarding policy may mitigate the effect of mid-flow route changes. Rather than dropping a privileged packet whose capability fails the verification test, a router can simply demote the privileged packet to unprivileged status. Furthermore, if unprivileged packets were marked in the same way as privileged packets (with markings pushed into the MSB of the capability field) then this mechanism would allow the demoted packet to carry the updated capability to the server in the same way that a privileged packet would. Using this scheme, route changes would only effect privileged connectivity when the server is under DDoS.

5.5 Multiple Capability Classes

In order to better accommodate sessions with different packet frequencies, routers can have multiple valid capabilities which change at different frequencies. The routers would decide which capability to insert (or verify) in the packet based on a special TOS-like field that would be initialized by the client based on its session requirements. The server is, of course, at liberty to refuse to respond to a client requesting a long-lived capability. However, a server may also delay the transmission of a long lived capability, to minimize its useful validity time at the client.

6 Conclusion

Today's Internet is susceptible to DDoS flooding and flash crowds, where network links are saturated upstream from a victim, causing the victim's service to become unavailable to its clients. TCP services are particularly vulnerable to such flooding attacks, as the TCP exponential backoff mechanism causes a severe reduction in performance if packet loss is above 5%. These flooding vulnerabilities are possible in part because of the end-to-end principle, in which the network is modeled simply as a transit mechanism for packets, and all interpretation of those packets is meant to take place at the packet's destination. However, packet floods disable the network as they converge on links upstream of a victim, and there is no "intelligence" in the network to filter them out.

Some research has been done into DDoS flooding countermeasures, however these solutions often require per-flow state on routers, inter-ISP collaboration, an overlay infrastructure, or extensive router processing.

In this paper, we present SIFF, a novel design that addresses the DDoS flooding problem in a future Internet setting, without relying on any of the above assumptions. Using this design as a basis, we also present a countermeasure that may be deployed in the current Internet, assuming that client and server software is updated. SIFF does not require any of the above assumptions of previous countermeasures. In SIFF, all network traffic is separated into privileged and unprivileged packets, with the goal of protecting privileged packets from unprivileged packet flooding, and allowing packet receivers to selectively terminate individual privileged flows and have their packets be dropped deep in the network, before arriving near the victim. Those clients and servers who deploy our protocol will see significant immunity to packet flooding, provided that their ISP deploys updated routers with our technology.

7 Acknowledgments

We would like to thank Tom Anderson, Hal Burch, David Maltz, Srini Seshan, Ion Stoica, Hui Zhang and the anonymous reviewers for their feedback and suggestions on how to improve this paper. We would especially like to thank Virgil Gligor and Vern Paxson for their insightful feedback as shepherds of this paper.

References

[1] Daniel Adkins, Karthik Lakshminarayanan, Adrian Perrig, and Ion Stoica. Taming IP packet flooding attacks. In *Proceedings of Workshop on Hot Topics in Networks (HotNets-II)*, November 2003.

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

[3] D. Scott Alexander, Kostas G. Anagnostakis, William A. Arbaugh, Angelos D. Keromytis, and Jonathan M. Smith. The price of safety in an active network. In *SIGCOMM '99*, 1999.

[4] David G. Andersen. Mayday: Distributed filtering for Internet services. In *Proceedings of USITS*, 2003.

[5] Tom Anderson, Timothy Roscoe, and David Wetherall. Preventing Internet denial-of-service with capabilities. In *Proceedings of Hotnets-II*, November 2003.

[6] S. Bellovin, M. Leech, and T. Taylor. The ICMP traceback message. Internet-Draft, draft-ietf-itrace-01.txt, October 2001.

[7] Robert Braden. Requirements for Internet hosts – communication layers. Internet Request for Comment RFC 1122, Internet Engineering Task Force, October 1989.

[8] Hal Burch and Bill Cheswick. Tracing anonymous packets to their approximate source. Unpublished paper, December 1999.

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

[10] CERT. TCP SYN flooding and IP spoofing attacks. Advisory CA-96.21, September 1996.

[11] CERT. Increased activity targeting windows shares. Advisory CA-2003-08, March 2003.

[12] Drew Dean, Matt Franklin, and Adam Stubblefield. An algebraic approach to IP traceback. *ACM Transactions on Information and System Security*, May 2002.

[13] Sven Dietrich, Neil Long, and David Dittrich. Analyzing ditributed denial of service attack tools: The Shaft case. In *14th Systems Administration Conference, LISA 2000*, 2000.

[14] Dave Dittrich. Distributed Denial of Service (DDoS) attacks/tools resource page. `http://staff.washington.edu/dittrich/misc/ddos/`, 2003.

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

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

[17] Virgil Gligor. On denial of service in computer networks. In *Proceedings of International Conference on Data Engineering*, pages 608–617, February 1986.

[18] Virgil Gligor. Guaranteeing access in spite of service-flooding attacks. In *Proceedings of the Security Protocols Workshop*, April 2003.

[19] Michael Goodrich. Efficient packet marking for large-scale IP traceback. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 117–126. ACM Press, 2001.

[20] Christian E. Hopps. Analysis of an equal-cost multi-path algorithm. Internet Request for Comment RFC 2992, Internet Engineering Task Force, November 2000.

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

[22] 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)*, February 2002.

[23] Hani Jamjoom and Kang G. Shin. Persistent dropping: An efficient control of traffic aggregates. In *Proceedings of ACM SIGCOMM '03*, pages 287–297, August 2003.

[24] 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.

[25] A. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure Overlay Services. In *Proceedings of ACM SIGCOMM*, August 2002.

[26] 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.

[27] 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.

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

[29] Jelena Mirkovic, Gregory Prier, and Peter Reiher. Attacking DDoS at the source. In *ICNP*, 2002.

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

[31] 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. USENIX.

[32] 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.

[33] Vern Paxson. End-to-end routing behavior in the internet. In *ACM SIGCOMM Computer Communications Review*, volume 26, pages 25–38, October 1996.

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

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

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

[37] 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.

[38] Dawn Song and Adrian Perrig. Advanced and authenticated marking schemes for IP traceback. In *Proceedings IEEE Infocomm 2001*, April 2001.

[39] Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker, and Sonesh Surana. Internet Indirection Infrastructure. In *Proceedings of ACM SIGCOMM 2002*, pages 10–20, August 2002.

[40] Ion Stoica, Scott Shenker, and Hui Zhang. Core-stateless fair queueing: A scalable architecture to approximate fair bandwidth allocations in high speed networks. In *SIGCOMM '98*, 1998.

[41] Ion Stoica and Hui Zhang. Providing guaranteed services without per flow management. In *Proceedings of ACM SIGCOMM'99*, pages 81–94, 1999.

[42] Robert Stone. CenterTrack: An IP overlay network for tracking DoS floods. In *Proceedings of the 9th USENIX Security Symposium*. USENIX, August 2000.

[43] 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.

[44] R. Teixera, K. Marzullo, S. Savage, and G. Voelker. Characterizing and measuring path diversity of internet topologies. In *SIGMETRICS'03*, June 2003.

[45] R. van den Berg and P. Dibowitz. Over-zealous security administrators are breaking the Internet. In *Proceedings of 2002 LISA Conference*, November 2002.

[46] XiaoFeng Wang and Michael K. Reiter. Defending against denial-of-service attacks with puzzle auctions. In *Proceedings of IEEE Symposium on Security and Privacy*, May 2003.

[47] Abraham Yaar, Adrian Perrig, and Dawn Song. Pi: A Path Identification mechanism to defend against DDoS attacks. In *Proceedings of IEEE Symposium on Security and Privacy*, May 2003.

A Fragmentation and the IP Identification Field

Placing a 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). Despite the fact that fragmented traffic represents only between 0.25% and 0.75% of packets in the Internet [36, 41], we consider a mechanism to allow packet marking to coexist with fragmentation.

We offer the solution 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 [45], as specified by the Path MTU Discovery standard in RFC 1191 [30]. Packets that do not match this predicate are ineligible for being EXP packets. DTA packets could be fragmented, provided that all the fragments follow the same path through the network, and that the fragmenting router doesn't reset any of the reserved bits that are normally set in a DTA packet.

Unfortunately, the `DF` classification is inadequate for UDP traffic, of which a much smaller percent of traffic has the `DF` bit set. 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 [7], however, lower MTU links are possible and perhaps likely, with the expected proliferation of web-enabled phones. We show the percent of markable traffic from a 31 day trace of packets from the Lawrence Berkeley Lab DMZ in Table 2. In either case, the networking community will need to agree on a specific value for a minimum MTU before we can execute our handshake algorithm over non-`DF` or 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 2. Percent of packets that can be marked by classification. Average over 31 days of traffic from Lawrence Berkeley Lab DMZ, May 1-31, 2003.