On Filtering of DDoS Attacks Based on Source Address Prefixes

Gary Pack Jaeyoung Yoon Eli Collins Cristian Estan
Computer Sciences Department
University of Wisconsin-Madison
{*pack,jyoon,eli,estan*}*@cs.wisc.edu*

Abstract

Distributed denial of service (DDoS) attacks are a grave threat to Internet services and even to the network itself. Widely distributed "zombie" computers subverted by malicious hackers are used to orchestrate massive attacks. Any defense against such flooding attacks must solve the hard problem of distinguishing the packets that are part of the attack from legitimate traffic, so that the attack can be filtered out without much collateral damage. We explore one technique that can be used as part of DDoS defenses: using ACL rules that distinguish the attack packets from the legitimate traffic based on source addresses in packets. One advantage of this technique is that the ACL rules can be deployed in routers deep inside the network where the attack isn't large enough to cause loss of legitimate traffic due to congestion. The most important disadvantage is that the ACL rules can also cause collateral damage by discarding some legitimate traffic. We use simulations to study this damage how it is influenced by various factors. Our technique is much better than uninformed dropping due to congestion, but it produces larger collateral damage than more processing-intensive approaches. For example it can reduce the attack size by a factor of 3 while also dropping between 2% and 10% of the legitimate traffic. We recommend the use of source address prefix based filtering in combination with other techniques, for example as a coarse pre-filter that ensures that devices performing the processing-intensive filtering are not overwhelmed.

1 Introduction

Distributed denial of service (DoS) attacks are a major threat to the reliable functioning of Internet services and current measures against them, while effective in some instances, have not been sufficient to eradicate this threat. Moore et al.[22] identified more than 4,000 attacks per week using a conservative method that underestimates the number of attacks. While most attacks are short and target small sites, large well-provisioned sites are far from immune from this threat. There are reports of large attacks with between 600,000 and 1,800,000 packets per second or more [22, 11, 31] and data volumes as high as 3 Gbits/s [4]. These statistics are for attacks from 2003 and earlier; today's attacks are likely larger. (D)DoS attacks have disrupted large search engines, e-commerce sites, news sites [12], and root DNS servers [31]. The threat of floods has been used repeatedly by various criminal organizations to extort "protection money" from businesses with an online presence [26, 4, 6]. A disturbing development is that in the last few years malicious hackers have launched DDoS attacks using large networks of "zombies": computers taken over through a worm or through some other automated method. The Code Red II worm infected 359,000 computers [21] and an earlier version of that worm has been programmed to perform a DDoS attack against www1.whitehouse.gov. The sheer size of observed zombie networks together with the fact that many of the zombie computers have high speed Internet connections gives us reason to fear that future attacks could be more vicious than what we have witnessed so far and their effects even more crippling.

In this paper we investigate a light-weight approach to filtering attack traffic based on the historic distribution of packet source addresses arriving at a given IP address and service port. We hypothesize that the distribution of source addresses is relatively stable over a period of days for some services and weeks for other services. The results of our measurements are consistent with this hypothesis. Furthermore, the distribution of source addresses in the flood can differ significantly from the historic distribution of clients for the server under attack. We then use these distributions combined with examples of current traffic to generate prefix filtering rules that allow as much traffic through as possible so as to nearly fill the capacity of the link. An even stronger defense would combine source address prefix filtering (SAPF) with a "traffic scrubbing" solution that performs more complex processing to distinguish between legitimate and illegitimate traffic. The goal is to avoid congesting the bottleneck link (or overwhelming the traffic scrubbing device) and at the same time allow as much legitimate traffic through as possible. We refer to any legitimate traffic that is filtered out as collateral damage.

The main advantage of our technique is that it can be applied against the largest floods as ACLs are supported by even the highest speed routers, and they can be "pushed out" deep enough into the network where the attack is not large enough to cause congestion. The computational requirements to derive the ACL rules are small, and a relatively small number of rules is required, 20 to 50 in our experiments. These computations are based on widely supported traffic data such as sampled NetFlow that can be collected without slowing the routers down. The ISP can deploy SAPF against multiple simultaneous attacks targetting different clients, as the ACL rules also contain the address of the victim in the destination prefix field. The main disadvantage of SAPF is that it can produce significant collateral damage. Therefore it is best used as a pre-filter to more processing-intensive defenses, or in combination with other techniques.

2 Related work

Defending against distributed denial of service attacks is an important problem that has the attention of the academic community and industry alike. We divide the related work into three distinct categories: detection of DoS floods, tests to distinguish attack traffic from legitimate traffic, and complete solutions to the DoS problem. The difference between the second category and the first is the focus on filtering out the attack traffic. The difference between the second and the third category is less well defined. But we consider a piece of work to be in the second category if its main contribution is to propose a good test for differentiating attack traffic from legitimate traffic, and in the third if it proposes a comprehensive solution to the DDoS problem. Our work fits into the second category.

2.1 Detecting DoS attacks

A first step in defending against a flood is to detect that an attack is in progress and to identify the victim(s). MULTOPS [10] allows routers to detect the victims of flooding attacks by tracking inbalances between the two directions of traffic using a data structure that adapts to the current distribution of destination addresses. Jung et al. use mappings from IP addresses to AS numbers [14] to distinguish between flooding attacks and flash crowds.

2.2 Differentiating between the attack and the legitimate traffic

Some denial of service attacks achieve their goals with relatively little traffic by exploiting protocol weaknesses or vulnerabilities in implementations. SYN floods exhaust the memory of servers that allocate per connection state in response to SYN packets. Other attacks exploit (no longer common) bugs in Windows that causes the victim to hang or reboot when it receives certain malformed IP fragments. Defenses against such attacks work by recognizing and discarding the packets or packet sequences crafted to cause damage. The focus of this paper is not on such attacks but on brute force attacks that cause damage by producing severe congestion on the links connecting the victim to the Internet.

Floods can cause damage irrespective of the contents and the headers of their packets. Yet, packets that are part of the flood can be easy to distinguish. Starting with the earliest DDoS tools, floods of ICMP and UDP packets have been a popular weapon [19]. It is relatively easy to defend against such attacks if they are directed at a web or mail server if one can install a few ACL rules at uncongested high speed routers instructing them to drop the attack traffic. When the flood packets are not that easily distinguishable (a flood of TCP packets with destination port 80 against a web server), filtering them out is harder. "Blackholing" the IP address of the victim using the routing protocol to instruct all routers to drop traffic sent to the victim promptly stops the attack. The problem with this approach is that the routers also drop all the legitimate traffic to the victim[1]. Often the attackers spoof the source addresses of the packets in the flood to make it harder to filter the attack. Many defenses rely on identifying and filtering out the spoofed packets. Egress filtering [17] often implemented using uRPF BGP source filtering uses knowledge of the network topology to drop many of the spoofed packets close to the sources of the attack. Park and Lee show [27] that filtering based on routing information available to routers can be very effective against spoofed traffic in Internet-like topologies if deployed by as few as 20% of ISPs. The Spoofer project [5] estimates that despite such measures, one quarter of the computers in the Internet can still spoof source addresses. Close to the victim, TTL based filtering proposed by Jin et al. [13] can be applied to detect spoofed packets. This approach can filter out up to 90% of the flood without much collateral damage, but it requires custom equipment since it is not supported by current routers. Peng et al. [28] proposed distinguishing between attackers and legitimate clients based on source IP addresses, but they do not use prefixes, but store individual IP addresses in a Bloom filter requiring custom hardware. Based on their evaluation of the effectiveness of their technique it is hard to compare it with the one we propose here.

Since early 2005 floods that are harder to filter have been reported. As large zombie networks have at least tens of thousands of computers, attackers started using the real IP addresses of the zombies to initiate "legitimate" sessions that overload the server with large volumes of traffic [30]. Tests that rely on detecting spoofing cannot

[1] Recently Steven Bellovin proposed an extension to the IP protocol known as the "evil" bit[3]. Filtering on this bit could result in a major reduction of collateral damage, but since this extension is not (yet) widely supported by routers and operating systems, and the incetives for its adoption by malicious software have not (yet) been convincingly articulated, we do not consider it in this paper.

defend against such attacks. Kandula et al. [15] propose using CAPTCHAs (challenging clients to type a word shown in an image) to distinguish humans sending requests to a web server from automated programs flooding it with requests. This solution can defend against DDoS attacks with little collateral damage, and it is especially effective against "uplink" attacks that try to congest the path packets take from the server to the Internet. This solution does not generalize to non-interactive services such as DNS and email and it cannot protect the server from "downlink" attacks that congest the links bringing client requests to the server.

2.3 Complete DDoS solutions

Some of the proposed DDoS defenses take more radical steps to provide a definitive solution. Extending the Internet architecture with capabilities [32, 33] makes it impossible for the attackers to send large floods because routers check the capabilities in the packets and drop all traffic not authorized by the receiver. The SOS proposal [16] takes a different approach: hiding the server behind an overlay network so that the attacker cannot find out the actual address of the server and thus cannot direct a flood at it.

Some proposals for filtering out DDoS attacks advocate filtering close to the sources [20, 2]. These solutions can achieve good filtering with small collateral damage, but deployment of such solutions is hindered by a misalignment of incentives: the networks with the zombies spend on defenses and the potential victims benefit. Pushback [18] is an architecture for defending against DDoS attacks, but it does not address the problem of differentiating between flood packets and legitimate packets. The technique we propose is complementary and it could be integrated into a pushback-type architecture.

There are numerous commercial solutions implementing DDoS filtering at the victim or the victim's ISP [24, 23, 7, 9] (see Appendix B of [19] for a survey of commercial DoS Defenses). These solutions use one or both of the following approaches: filtering the traffic at high speed routers using ACL rules derived from measurements of the attack traffic, and running the traffic through "traffic scrubbing" devices placed between the server to be protected and the rest of the Internet. Individual traffic scrubbing appliances can be overwhelmed by very large attacks. Agarwal et al. propose building regional centers with many such appliances within the ISP [1] and redirecting the traffic of servers under attack through these centers. Prolexic [29] uses a similar approach based on redirection to a high bandwidth data center that performs traffic cleaning using custom tools. Public material describes the architecture that allows traffic scrubbing devices to handle relatively high volumes of traffic by using specialized hardware, but there aren't many details about the tests used to distinguish between attacks and legitimate traffic. We found no indication that any of these solutions do source prefix based filtering as proposed in this paper, and we believe that these solutions might be improved through the use of the technique we present.

3 Source address prefix based filtering of DDoS floods

The goal of SAPF is to apply ACL rules at routers that drop enough traffic to keep the remaining traffic volume within a target rate. To compute these ACL rules we start with two sets of sampled flow records. The first has the entire traffic of the victim during a non-attack period which we use as a "baseline" description. The other set of records has the traffic sent to the victim during an ongoing attack. The filtering algorithm generates a list of ACL rules for routers at the victim's ISP that filter *the traffic sent to the victim* based on source IP prefixes. The filtering algorithm tries to limit the collateral damage (legitimate traffic dropped) and the number of rules kept within a pre-specified budget. ACL rules are recomputed repeatedly throughout the attack so that the filtering can adapt to changes in traffic.

Our measure of the effectiveness of a set of ACL rules is the amount of legitimate traffic dropped, measured in bytes. We could use other metrics such as the number of blocked IP addresses that send legitimate traffic, or something that differentiates between important clients and unimportant ones. Ideally we should minimize the monetary loss due to collateral damage. While the number of bytes of legitimate traffic discarded is not an exact measure of monetary loss, we consider it a better approximation than the number of legitimate IP addresses blocked. For example there can be many clients behind a large web proxy and thus blocking that proxy inflicts more damage than blocking a single user not using a proxy.

We impose two limitations on the sets of ACL rules we consider: the traffic that passes should not exceed the target rate and the number of rules should be below a pre-specified threshold. The number of ACL rules routers can support is limited by hardware resources (e.g. the size of the TCAM used to implement packet classification). Furthermore, these rules are used for purposes other than incident response [8], so the number of rules we can use for filtering out DDoS attacks is significantly smaller than hardware limits. And if there are multiple attacks active at the same time, the hardware has to support separate ACL rules for each attack because each rule will have in the destination field the IP address (or prefix) of the victim it protects. In Section 4 we show that, while a larger number of ACL rules can potentially improve protection, too many rules can cause overfitting in the algorithm computing the ACL and thus increase collateral damage. In our experiments we typically limit the number of rules to 100.

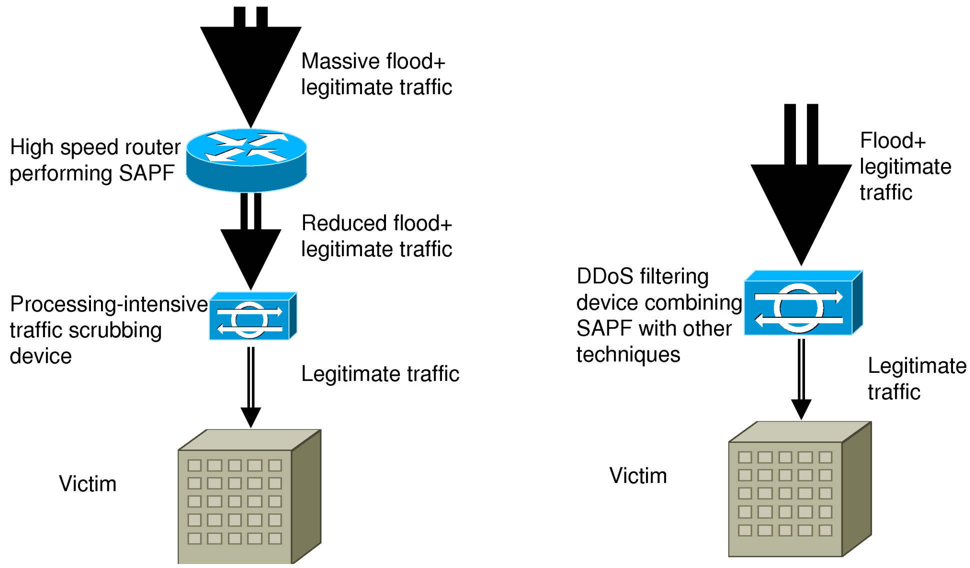

Figure 1: Source address prefix filtering is a technique best used as part of broader DDoS defenses. The example of the left uses SAPF to do a coarse filtering of a massive flood and a processing-intensive traffic scrubbing device to separate the remaining attack traffic from the legitimate traffic. The example on the right shows how SAPF can be used in combination with other tests to separate the flood from the legitimate traffic.

3.1 Evading SAPF

Source address prefix filtering (SAPF) can only be effective if the addresses that most legitimate traffic comes from do not appear often as source addresses in attack traffic. Since attackers can spoof source addresses, what keeps them from exactly matching the distribution of source addresses in legitimate traffic? There are two main reasons why SAPF can be helpful despite attackers trying to mimic the distribution of the sources address of legitimate clients. First, the attackers are not likely to have an accurate description of the typical legitimate traffic for the server; second, spoofing sources to mimic the legitimate traffic exposes the attackers to other countermeasures such as TTL based filtering [13]. For the attacker to get the exact distribution of clients, she would have to break into the server itself or into another computer used for storing its logs or the flow records collected by the first-hop router. SAPF cannot protect against such attackers, but if the attacker can break into the victim, she can probably cause significant damage without resorting to a flooding attack. In some sense the distribution of the source address of legitimate clients is the secret key that allows SAPF to protect the victim to some extent from the attackers.

In our experimental evaluation of collateral damage we consider the two strategies widely employed by current tools: spoofing source addresses at random from the routable unicast address space, and using the actual addresses of the zombies. We also consider attacks that try to mimic the legitimate traffic. For these attacks we assume that the attacker has the logs of servers other than the victim and uses them as a model for the distribution of source addresses in the flood: each source address present in the legitimate traffic of the model server will appear in the attack traffic, and it will represent the same percentage of both types of traffic. The collateral damage caused by filtering such attacks depends on the similarity between the client populations of the two servers and this is influenced not just by the type of server (web, DNS, email, etc.) but also by the type of information the two servers are hosting and how much overlap there is between the sets of users interested in it. Section 4.4 has the full details of the experimental setup and a discussion of our results.

3.2 SAPF as part of broader solutions

We do not consider source address prefix based flood filtering a complete solution, but an imperfect test for distinguishing legitimate traffic from some types of attack traffic. SAPF is best used in combination with other techniques as part of broader DDoS defenses (see Figure 1). One possibility is to use SAPF in combination with traffic scrubbing approaches when the attack is larger than the capacity of the available traffic scrubbing appliance: high speed routers inside the ISP could filter out some of the traffic directed at the victim, while the appliance would apply its filtering to the remaining traffic. For attacks large enough to congest the ISP's links close to the victim SAPF can be applied at multiple routers, deeper in the network. Another possibility is to combine the source prefix tests with other techniques to arrive at a better filtering solution. For example imagine a DDoS protection solution that does flow reassembly to detect floods based on application-level clues that needs to time out connection records to avoid running out of memory. When combined with SAPF, such a solution could be more aggressive in timing out and eventually blocking connections that come from suspicious prefixes. We do not evaluate SAPF in combination with other methods in this paper. Our goal is to give a quantitative answer to the question of how good source prefix information is at separating flood traffic from legitimate traffic.

When deployed as part of DDoS defenses, SAPF would be extended with various tuning knobs. For example the network administrator might forbid the algorithm from filtering out some important, low volume customers whom the algorithm could discriminate against. He might also assign weights to different prefixes of client addresses, to bias our algorithms towards protecting more important clients. Another possibility is to allow the network administrator to manually modify or override the ACLs generated by SAPF. In this paper we only evaluate SAPF as a defense operating on its own through ACL rules installed in a single high speed router.

3.3 Algorithms generating ACL rules

In this section we briefly discuss the algorithms we propose for generating the ACL rules. The technical report version of this paper [25] gives a more detailed description. Our algorithms generate the filtering rules based on a comparative analysis of the traffic at the time of the attack with regular traffic during an earlier "baseline" period. There are $2^{33} - 1$ possible prefixes, and the number of combinations of prefixes we could use is many orders of magnitude larger. We do not advocate performing an exhaustive search in this space for the optimal list of ACL rules, but the use of simpler, faster, greedy heuristics. We experiment with three types of algorithms, producing three types of outputs: the "positive" algorithm denies all traffic going to the victim with the last (default) rule in the list and the other rules specify non-overlapping source prefixes that are allowed to pass; the "negative" algorithm allows all traffic by default and the list contains rules for specific non-overlapping prefixes that should be filtered out; the "mixed" algorithm gives a list with a mix of "accept" and "deny" rules with possibly overlapping prefixes arranged so that the more specific prefixes come before the more general ones.

To simplify the choice of prefixes in the output, all three algorithms cluster the traffic. Once we have clustered the traffic, the algorithms choose prefix lengths and determine which prefixes to add to the output list and which not to (and the mixed algorithm also needs to decide whether to allow or deny the prefixes it adds to the output). This stage ensures that the number of rules is within the budget and that the total traffic that passes the filter does not exceed the target rate. In picking which prefixes to allow and which to deny, the algorithms make simple greedy choices: prefixes that send much traffic during the flood but not much in the baseline period are denied, and prefixes that send much traffic in the baseline but are a smaller percentage of the traffic during attack are allowed.

4 Experimental evaluation

The usefulness of SAPF depends on the extent of the collateral damage it inflicts. As discussed in Section 3, we measure collateral damage as the percentage of the legitimate traffic (in bytes) that is filtered out. We use actual NetFlow records to reconstruct the legitimate traffic of various servers traffic and we mix in controlled synthetic DDoS floods to evaluate the amount of collateral damage in various scenarios. We look at how the following factors affect collateral damage: the size of the attack, the strategy the attackers use to mimic legitimate traffic, the degree of overprovisioning, the number of ACL rules used, the choice of filtering algorithm and the choice of "baseline" period used as description of the legitimate traffic.

4.1 Methodology and description of data

We use two sets of NetFlow data for our experiments. The first set has traffic coming from the Internet into our university's campus over two one-week periods separated by a month in the first half of 2005. The second data set represents the traffic for June 2005 on an OC12 peering link of a regional ISP. The Campus data was collected with a sampling rate of 1 in 256 packets and the ISP data with 1 in 10 packets. We simulate an 8 hour attack during the busiest time of a Thursday, from 9:00 AM to 5:00 PM. For each server type we pick the one with the most traffic in its trace as a victim of the simulated DDoS attack. For the ISP data we use the largest SMTP server (peak traffic 74.4 MB/5 minute bin), the largest HTTP server (peak traffic 4.67MB/5 minute bin)[2], and the largest DNS server (peak traffic 1.36 MB/5 minute bin). For the Campus data set we use the largest SMTP server (peak traffic 95.6 MB/5 minute bin), the largest HTTPS server (peak traffic 40.7 MB/5 minute bin), and the largest DNS server (peak traffic 1.28 MB/5 minute bin). We do not consider the largest web server from the Campus data set because it experienced an apparent outage during the period of the simulated attack.

During the attack, we re-run the algorithms for generating the ACL rules every 5 minutes. In an actual deployment the available measurement data about an attack would be a few minutes old. To capture this disadvantage

[2]Note that this is the direction of traffic going towards the web server, traffic in the opposite direction is much larger.

in our experiments, we evaluate the collateral damage inflicted by the filtering rules on the *next* 5 minutes' traffic, not on the traffic the rules were computed for. For each setup, we have 95 data points for the 5 minute bins in the 8 hour period of the attack when filtering of the attack takes place.

For the attack traffic we use three methods for generating the distribution of source addresses corresponding to the three strategies discussed in Section 3.1. For attacks with source addresses spoofed at random we use a uniform distribution of 100,000 random IP addresses from the unicast address space (except unroutable prefixes 0.0.0.0/8, 10.0.0.0/8, 127.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16). For attacks where zombies use their own IP addresses we need to model the distribution of addresses for the zombies. We do not have data that allows us to measure directly such distribution, so we used an indirect method instead. Zombies are often subverted through worms, and computers infected by worms continue to scan. By logging scans to unused address space we can get an approximate distribution of the computers infected by worms and we use it to model the distribution of zombies. We used a one week trace of scans captured by two unused /19 campus networks. The third attack strategy we consider tries to mimic the distribution of legitimate clients of the victim by using the traffic of another server as a model. We use as models other servers from our data sets. For all types of attacks we control the volume of the attack by controlling the amount of traffic sent by individual source addresses.

4.2 A first look at collateral damage

In subsequent experiments we look at the effect of various factors on the size of the collateral damage. The first experiment discussed here uses the default values for the various factors. We assume a link capacity (or target rate) of 2 times the peak traffic of the victim, a flood volume of 5 times the peak traffic of the victim, an attack that uses the actual addresses of the zombies, a limit of 100 on the number of ACL rules, and we use the positive algorithm with the previous 3 days' traffic as baseline.

We can quantify the reduction in collateral damage we achieve by using filtering rules that discriminate against attack traffic and favor legitimate traffic. To do this we compare against a simple "uninformed filtering" algorithm that drops legitimate and attack connections with the same probability. For the default configuration the total traffic is 6 times the peak legitimate traffic, whereas the link capacity is 2 times peak legitimate traffic, so two thirds of the traffic is dropped. Note that uninformed filtering is better than the behavior of routers which drop packets indiscriminately because their queues are full: under the current behavior of routers, congestion aware TCP compliant legitimate clients reduce their sending rate whereas the attackers don't and they end up using the entire bandwidth.

Victim server	Collateral damage 5th percentile/avg./95th percentile	
	Src. addr. filtering	Uninformed filt.
ISP Mail	0.4 / 2.6 / 12.7%	62.8 / 64.0 / 65.7%
ISP Web	2.4 / 8.3 / 17.0%	64.6 / 66.1 / 67.3%
ISP DNS	7.0 / 9.6 / 12.8%	66.9 / 67.7 / 68.3%
Campus Mail	0.9 / 4.1 / 14.5%	62.7 / 64.3 / 66.2%
Campus HTTPS	0.0 / 1.9 / 6.8%	62.8 / 63.9 / 65.7%
Campus DNS	0.0 / 8.6 / 16.9%	64.4 / 65.8 / 67.4%

Table 1: Collateral damage with the default parameters. Positive and uninformed.

Victim server	Algorithm	
	Mixed	Negative
ISP Mail	0.2 / 3.0 / 17.4%	1.2 / 15.1 / 35.6%
ISP Web	9.3 / 17.5 / 29.0%	4.6 / 12.6 / 22.9%
ISP DNS	6.9 / 10.0 / 13.9%	22.1 / 36.2 / 48.4%
Campus Mail	0.2 / 6.0 / 25.4%	3.7 / 16.3 / 36.8%
Campus HTTPS	0.1 / 6.7 / 26.6%	2.3 / 7.2 / 23.8%
Campus DNS	1.2 / 6.8 / 15.4%	16.1 / 33.0 / 57.8%

Table 2: The collateral damage with mixed and negative algorithms for default parameters.

Table 1 and 2 show that for all victims, SAPF positive and mixed algorithms have around an order of magnitude lower collateral damage than uninformed filtering. SAPF is more effective protecting SMTP and HTTPS servers than HTTP and DNS servers which incur larger collateral damage. The reason is that the distribution of legitimate clients for HTTP and DNS servers is more similar to distribution of zombies. We note that the HTTPS server had by far the fewest distinct client IP addresses of all servers (149 per week in the campus NetFlow data sampled 1/256 compared to 21,384 for DNS and 59,017 for SMTP), and it was easier to protect such a small client population.

For bins where the legitimate traffic is lower than its peak, the collateral damage of uninformed filtering is slightly below 66.7%. For other bins it is slightly higher. The reason is that we rounded down the peak traffic to the next megabyte when computing the link capacity and attack size.

Our most important conclusions are as follows.

- SAPF is an order of magnitude more effective than uninformed filtering.
- Some types of servers (SMTP) are significantly easier to protect than others (web, DNS).

4.3 Intensity of attack and degree of overprovisioning

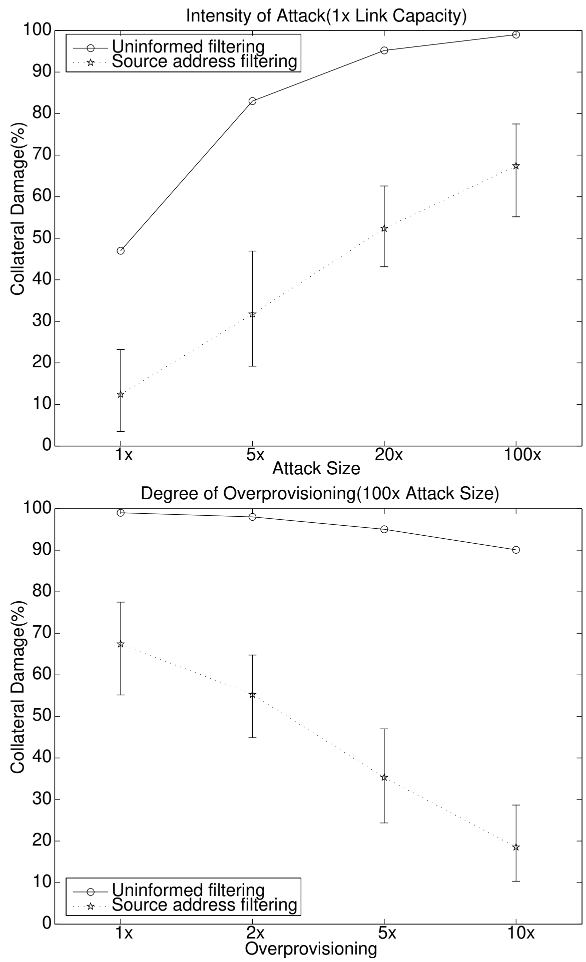

Figure 2: The effect of increased attack sizes and increased overprovisioning for the ISP web server.

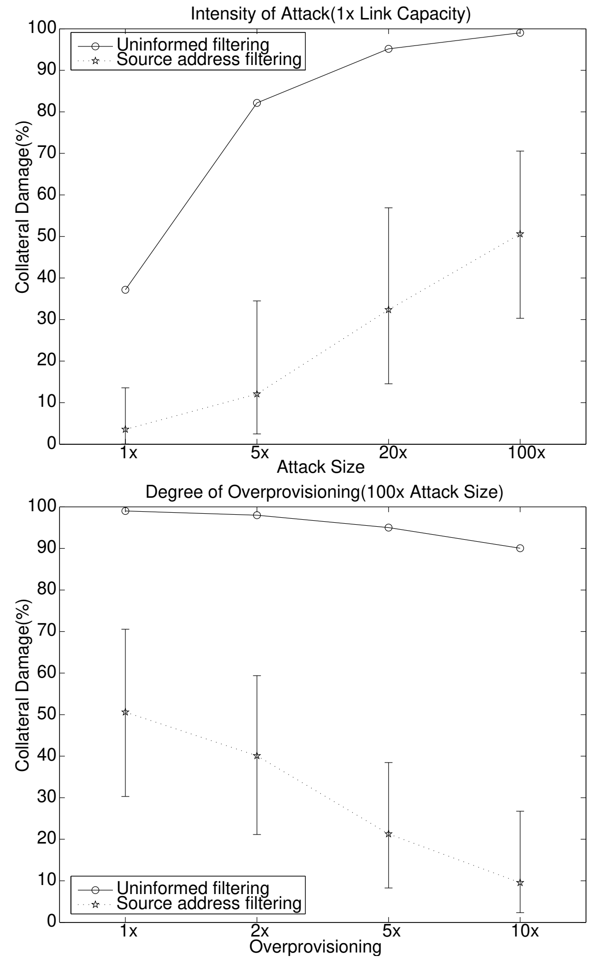

Figure 3: The effect of increased attack sizes and increased overprovisioning for the Campus mail server.

The intensity of the attack and the size of the link connecting the server to the Internet influence how aggressive the filtering has to be to keep the link uncongested. Here we present experiments evaluating the effect of those two factors on the collateral damage. Figures 2 and 3 show result for attacks against the ISP web server and the Campus mail server. The first plot in each figure shows how the collateral damage increases as we increase the attack size from 1 times the peak legitimate traffic to 100 times, with a tightly provisioned link of capacity equal to the peak traffic. In second plot in each figure we increase the link capacity to up to 20 times peak legitimate traffic with constant attack size of 100 times peak traffic. Collateral damage decreases as we increase the linik capacity because we can afford to filter less aggressively. As in the previous section, the collateral damage is consistently lower for the mail server than for the web server. The advantage of SAPF over uninformed filtering is even larger for the more challenging scenarios with large attacks against small links. Whereas uninformed filtering would allow 1% of the legitimate traffic through when the attack size is 100 times the link capacity, SAPF allows 49.4% of the legitimate traffic to pass for the mail server and 32.6% for the web server.

The higher the ratio between flood size and link size, the larger the collateral damage. But even when we keep this ratio constant, the collateral damage depends on the attack size. Table 3 has in its first column the collateral damage of an attack 20 times larger than the peak traffic against a link only 2 times the peak traffic and in the second column an attack 100 times the peak traffic against a link 10 times the peak traffic. For all servers the collateral damage is significantly lower for the second scenario. We conclude that SAPF could be a good pre-filter in systems using multiple filters against very large attacks.

Our most important conclusions are as follows.

Victim server	Collateral damage 5th percentile/avg./95th percentile	
	Flood 20x link 2x	Flood 100x, link 10x
ISP Mail	2.6 / 12.1 / 29.9%	1.7 / 8.3 / 24.2%
ISP Web	18.6 / 29.8 / 41.3%	10.3 / 18.5 / 28.7%
ISP DNS	27.3 / 30.6 / 34.1%	14.8 / 17.0 / 20.1%
Campus Mail	4.6 / 14.0 / 29.0%	2.3 / 9.5 / 26.7%
Cmps. HTTPS	1.6 / 5.8 / 13.6%	1.4 / 4.2 / 9.0%
Campus DNS	22.0 / 34.7 / 46.2%	14.7 / 25.8 / 36.3%

Table 3: Overprovisioning has a stronger effect than attack size.

Port number (service name)	Distinct addresses
445 (SMB)	339,047
135 (DCOM)	9,642
80 (Web)	4,333
6129 (Dameware)	2,318
1433 (Msft. SQL)	886

Table 4: Number of scanners.

- SAPF improves its advantage over uninformed filtering for large attacks.
- Overprovisioning has a stronger effect on collateral damage than the size of the attack.

4.4 Attack strategy

SAPF relies on finding differences between the distribution of source addresses of regular traffic and the flood, so the strategy the attacker uses to set the source addresses influences the effectiveness of filtering. We first compare two strategies implemented by existing tools: spoofing source addresses at random and using the actual source addresses of the zombies. Next we evaluate the effectiveness of trying to mimic the source address distribution of the victim's legitimate clients by modeling attacks after other servers' traffic. All attacks have the same volume (5 times the peak traffic of the victim). We do not model the effects of egress filtering on attacks that spoof source addresses.

We model the distribution of zombies after the IP addresses of computers scanning unused IP address space because these computers are often infected by worms, and thus likely to be turned into zombies. In practice there are many networks of zombies (botnets) controlled by different groups. We model different zombie networks by grouping scanners based on the port number they scan on, because computers with different vulnerabilities are likely to be infected by different worms (and subsequently controlled by the writers of those worms). Table 4 gives the sizes of the 5 networks of zombies used in this section. For all other experiments we used a sample of 100,000 IP addresses from those scanning on port 445.

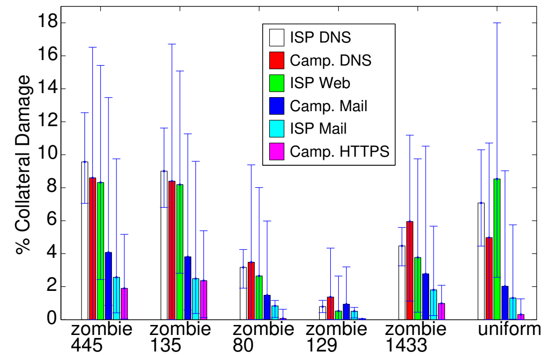

Figure 4: Attacks with different zombie networks using their own addresses or spoofing uniformly at random.

Figure 4 shows a comparison of the collateral damage inflicted by attacks using the IP addresses of the 5 zombie networks and uniformly spoofed source addresses on the 6 servers we consider. We draw a number of surprising conclusions from these results. Contrary to our expectation, spoofing source addresses uniformly inflicts collateral damage similar to the most effective zombie network distribution. We would have expected that uniform spoofing will use more IP addresses from ranges where none of the servers have legitimate clients and thus be easier to filter with little collateral damage. It was not surprising to see that the address distributions of some zombie networks were able to inflict more damage than others. But the amount of damage is not related to the size of the zombie network. Ports 445 and 135 expose vulnerabilities in desktop clients and the zombies and inflict similarly high collateral damage despite the fact that the number of distinct IP addresses in the port 445 zombie network is one order of magnitude higher. The next most damaging zombie network is that built by exploiting Microsoft SQL, an application that typically runs on servers located close to the clients. It is more damaging than the larger zombie network based on exploiting web server vulnerabilities and significantly more damaging than the zombie network built by exploiting vulnerabilities in remote computer management software which are closely clustered in a few networks (566 distinct /16s for Microsoft SQL versus 649 /16s for Dameware).

Figure 5 shows the effects of attacks where the distribution of source addresses is modeled after the trafic of another server. The top plot shows attacks against the largest ISP web server and the bottom graph shows attacks against the largest Campus mail server. Each cluster of bars represents attacks based on traffic derived from the four examples of each server type indicated on the horizontal axis. We also included the results for attacks mod-

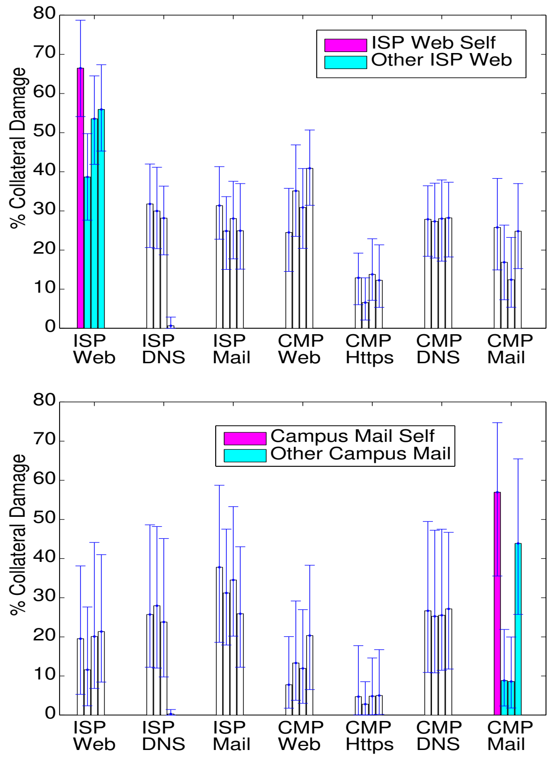

Figure 5: Attacks with source address distributions modeled after the traffic of servers of the same type as the victim can be very damaging.

eled after the victim's own traffic from an earlier week, and not surprisingly, the collateral damage inflicted by SAPF matched the damage of uninformed filtering. Other servers of the same type often proved to be damaging models (typically less damaging if from the other organization). Note that for attacks against the largest Campus mail server, two servers from the same campus proved poor models and the collateral damage of attacks modeled after them is similar to that of attacks not trying to mimic the distribution of legitimate clients. The DNS servers proved the most dangerous models other than servers of the same type as the victim, with DNS servers from the same organization slightly more damaging. A possible explanation is that clients to all services go through the DNS servers for address lookups, so the distribution of clients for the DNS servers is a combination of the distribution of the clients of all other services (more exactly their local DNS servers), but it does not give a good indication of the amount of traffic each legitimate client sends.

Our most important conclusions from comparing different strategies for *attacks with identical amounts of traffic reaching the victim* are as follows.

- Spoofing source addresses at random is as damaging as using the actual addresses in the most damaging of our simulated zombie networks.
- For all victims studied, the most damaging zombie networks are those that result from exploiting vulnerabilities in desktop clients, or servers close to them.
- The application exploited to build the zombie network is more important than the number of zombies.
- The damage caused by using SAPF on attacks modeled after other servers can be as high as the collateral damage with uninformed filtering, but the effect

4

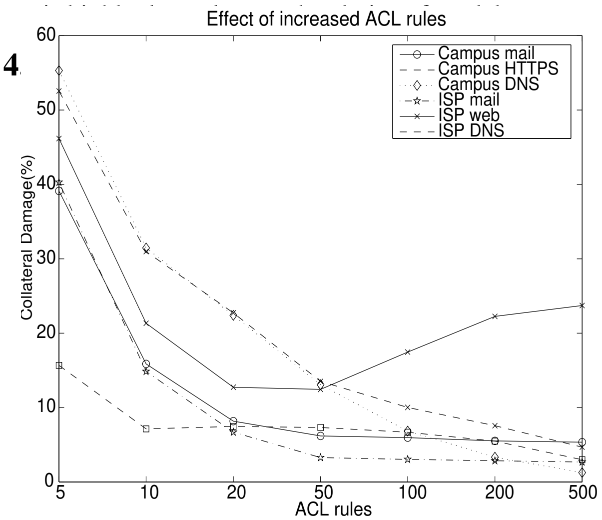

Figure 6: Typically collateral damage decreases as we increase the number of rules.

The number of ACL rules affects how exact our filtering of the traffic can be. With few rules we can do only coarse filtering, with more rules the granularity improves. Figure 6 plots the average collateral damage for the 6 servers considered when using the mixed algorithm to generate the filtering rules. For some servers the decrease in collateral damage is more pronounced as the number of rules increases. Surprisingly, for the ISP web server the collateral damage increases after we increase the number of rules above 50. By looking at the actual ACL rules generated we found the explanation of this apparent anomaly. With a larger budget for the number of rules, the filtering algorithm generates many ACL rules that deny individual IP addresses of legitimate clients which do not appear in the baseline, but for which there are other clients in the same /16 that do appear. Having only few ACL rules forces the algorithm to make decisions about few large aggregates. These larger aggregates do have traffic in the historic baseline, and are consequently not filtered out. But when the decision is made on individual IP addresses, the legitimate client address that is not in the baseline seems to be an attacker and it is filtered out. Improvements to the algorithms generating the ACL rules could eliminate this overfitting of the baseline data.

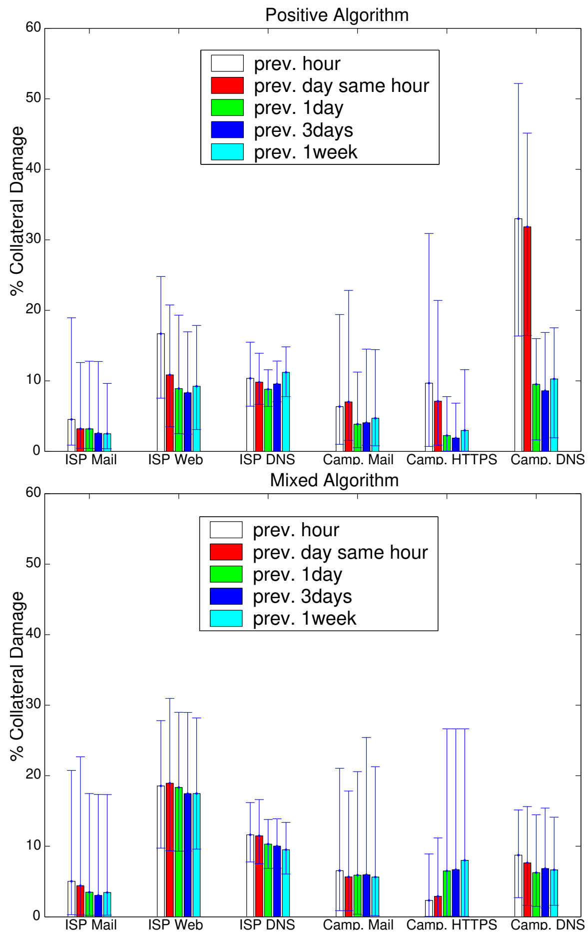

Figure 7: The effect of different baselines for the positive and mixed algorithm.

4.6 Baseline and filtering algorithm

We experimented with using various portions of the traffic log of the victim as baselines for the filtering algorithms. For most servers and most algorithms for generating ACL rules, the choice of the baseline period had a small effect on the collateral damage. Figure 7 shows that the 3 days prior to the attack worked well for all configurations. Using shorter baselines, closer to the time of the attack (the hour before the attack) does not capture the full diversity of legitimate clients. Using older baselines (an earlier week) can miss shifts in the client population yet in most instances these older baselines also work well.

We also compared our three algorithms “positive”, “mixed” and “negative” (see Section 3.3 for a description of the algorithms) with many values for the parameters. For brevity we omit the detailed results (see Table 2 for partial results). The positive and mixed algorithms have similar results for most configurations, but positive has a slight overall advantage. The negative algorithm results in significantly higher collateral damage than either of the other algorithms and we do recommend against its use.

5 Conclusions

Distributed denial of service attacks are an on-going concern for ISPs and companies with an online presence. Defending against flooding attacks is the subject of significant academic research efforts and there are many commercial solutions implementing DDoS defenses. A hard problem that all these defenses need to solve is distinguishing the attack traffic which should be filtered out from the legitimate traffic. In this paper we investigate a light-weight technique for filtering attack traffic based on the historic distribution of packet source addresses arriving at a given IP address and service port. The main advantage of this technique is that it can be applied at high speed routers without changes in hardware or software. The main disadvantage is the fact that it causes collateral damage. We performed extensive experimental evaluation to understand how various factors affect the collateral damage. SAPF is best suited to be used not as a sole defense, but as a pre-filter to more processing-intensive filters, or in combination with other techniques.

Our experimental evaluation of SAPF lead to a number of important conclusions. We evaluated three algorithms that use a comparative analysis of the traffic of the server during normal operation and the traffic during the attack to generate a short list of ACL rules that reduce the traffic to within a target rate. We used simulated attacks superimposed over traces of actual traffic to study the amount of collateral damage inflicted by source address prefix based filtering in various scenarios. This collateral damage is typically an order of magnitude lower than the damage incurred through uninformed filtering. The average damage is between 2% and 10% when the traffic during the attack is 3 times the target rate, with damage typically lower for mail servers than for DNS and web servers. For attacks using the actual addresses of the zombies, the effectiveness of SAPF increases if the zombie network is not built through a worm that exploits a desktop computer vulnerability. If the attacker mimics the distribution of the source address of legitimate clients of the victim by modeling the source addresses used in the attack after the clients of another server, the damage inflicted by SAPF can be close to that of uninformed filtering. Some servers, even some offering the same service as the victim, constitute a bad model for the attack and attacks modeled after their client distribution do not inflict more collateral damage than attacks with the actual source addresses of the zombies. The collateral damage inflicted by filtering increases with the attack size, and decreases as the capacity of the bottleneck link increases. Overprovisioning has a stronger positive effect than the negative effect of attack size. We have shown that the distribution of source addresses is relatively stable over periods of time sufficient to construct ACL rules which filter enough traffic to avoid congestion and allow a majority of the legitimate traffic to pass. We have also demonstrated a relative insensitivity of SAPF to baseline choice. From the experimental results we conclude that source address prefix based filtering can improve DDoS defenses.

6 Acknowledgments

We wish to thank Paul Barford and Vinod Yegneswaran for providing us with the logs of worm-infected IP addresses that we use to model the address distribution of zombie networks, Dave Plonka for collecting the NetFlow data for on-campus servers, and Jason Malacko for providing access to the ISP traffic data we used.

References

[1] Sharad Agarwal, Travis Dawson, and Christos Tryfonas. Ddos mitigation via regional cleaning centers. Technical report, Sprint ATL Research Report RR04-ATL-013177, August 2003.

[2] Katerina Argyraki and David R. Cheriton. Active internet traffic filtering: Real-time response to denial-of-service attacks. In *USENIX Technical Conference*, April 2005.

[3] Steven Bellovin. The security flag in the ipv4 header. RFC 3514, April 2003.

[4] Scott Berinato. How a bookmaker and a whiz kid took on an extortionist and won. `http://www.csoonline.com/read/050105/extortion.html`, May 2005.

[5] Robert Beverly and Steve Bauer. The spoofer project: Inferring the extent of internet source address filtering on the internet. In *USENIX SRUTI*, July 2005.

[6] Cassell Bryan-Low. Hacker hitmen. `http://www.wsjclassroomedition.com/archive/05feb/onln_hacker.htm`, February 2005.

[7] Cisco. Guard XT. `http://www.riverhead.com`.

[8] Cisco. Preparation for the attack: securing the network and the data plane. `ftp://ftp-eng.cisco.com/cons/isp/security/ISP-Security-Bootcamp-Singapore-2003/G-Preparation-DataPlane-ACLs-v3-0.pdf`.

[9] Cloudshield. Service provider managed security services. `http://www.cloudshield.com/what_we_do/Arbor_peakflowSP_OVW.html`.

[10] Thomer M. Gil and Massimiliano Poletto. Multops: a datastructure for bandwidth attack detection. In *USENIX Security Symposium*, July 2001.

[11] Patrick Gray. DDoS attack cripples Uecomm's AU links. `http://www.zdnet.com.au/news/security/0,2000061744,20273027,00.htm`, March 2003.

[12] Ann Harrison. Cyberassaults hit Buy.com, eBay, CNN and Amazon. `http://www.computerworld.com/news/2000/story/0,11280,43010,00.html`, February 2000.

[13] C. Jin, H. Wang, and K. Shin. Hop-count filtering: An effective defense against spoofed DoS traffic, 2003.

[14] Jayeon Jung, Balachander Krishnamurthy, and Michael Rabinovich. Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites. In *Proceedings of WWW*, May 2002.

[15] Srikanth Kandula, Dina Katabi, Matthias Jacob, and Arthur Berger. Botz-4-sale: Surviving organized ddos attacks that mimic flash crowds. In *NSDI*, May 2005.

[16] Angelos Keromytis, Vishal Misra, and Dan Rubenstein. SOS:secure overlay services. In *Proceedings of the ACM SIGCOMM*, August 2002.

[17] Tom Killalea. Recommended internet service provider security services and procedures. RFC3013, November 2000.

[18] Ratul Mahajan, Steven M. Bellovin, Sally Floyd, John Ioannidis, Vern Paxson, and Scott Shenker. Controlling high bandwidth aggregates in the network. *ACM SIGCOMM Computer Communications Review*, 32(3):62–73, July 2002.

[19] Jelena Mirkovic, Sven Dietrich, David Dittrich, and Peter Reiher. *Internet Denial of Service Attack and Defense Mechanisms*. Prentice Hall, December 2004.

[20] Jelena Mirkovic, Gregory Prier, and Peter Reiher. Attacking DDoS at the source. In *International Conference on Network Protocols*, November 2002.

[21] David Moore, Colleen Shannon, and Jeffery Brown. Code-red: a case study on the spread and victims of an internet worm. In *Internet Measurement Workshop*, 2002.

[22] David Moore, Geoffrey M. Voelker, and Stefan Savage. Inferring internet Denial-of-Service activity. In *10th USENIX Security Symposium*, 2001.

[23] Arbor Networks. Peakflow. `http://www.arbornetworks.com`.

[24] Mazu Networks. Mazu Profiler and Mazu Enforcer. `http://www.mazunetworks.net`.

[25] Gary Pack, Jaeyoung Yoon, Eli Collins, and Cristian Estan. On filtering of DDoS attacks based on source address prefixes. Technical Report 1547, University of Wisconsin-Madison, Department of Computer Sciences, December 2005.

[26] Denise Pappalardo and Ellen Messmer. Extortion via DDoS on the rise. Network World, May 2005.

[27] Kihong Park and Heejo Lee. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets. In *Proceedings of the ACM SIGCOMM*, August 2001.

[28] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. Protection from distributed denial of service attack using history-based ip filtering. In *IEEE International Conference on Communications*, pages 482–486, May 2003.

[29] Prolexic. `http://www.prolexic.com/`.

[30] The prolexic zombie report. `http://www.prolexic.com/zr/`, 2005.

[31] Paul Vixie, Gerry Sneeringer, and Mark Schleifer. Events of 21-oct-2002. `http://d.root-servers.org/october21.txt`, November 2002.

[32] 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*, May 2004.

[33] Xiaowei Yang, David Wetherall, and Thomas Anderson. A DoS-limiting network architecture. In *Proceedings of the ACM SIGCOMM*, August 2005.