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. Despite significant research efforts and the existence of a wide range of commercial products defending against them, DDoS attacks are still a concern for most network operators and companies relying on the Internet. A particularly hard problem is distinguishing the packets that are part of the attack from legitimate traffic so that the attack can be filtered out without much collateral damage. In this paper we explore the use of ACL rules that distinguish the attack packets from the legitimate traffic based on prefixes derived from models of the historic distribution of legitimate packet source addresses. One advantage of this defense is that these ACL rules can be deployed in routers deep in the network where the attack isn't large enough to cause loss of legitimate traffic due to congestion. The most important disadvantage is that these ACL rules can also cause collateral damage by discarding some legitimate traffic. We use simulations to study this damage. We examine the effect of various factors: magnitude of attacks, attack strategy, degree of network overprovisioning, number of ACL rules used, service targeted (web, email, DNS), and algorithm for generating ACL rules. For attacks 100 times larger than the link capacity provisioned to match peak traffic we applied SAPF to reduce the total traffic to within link capacity and it discarded on average 54% of the legitimate traffic for a mail server and 67% for a web server. For smaller attacks of only 5 times the link capacity the collateral damage was 8% and 31% respectively.

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.[21] 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 [21, 10, 28] and data volumes as high as 3 Gbits/s [3]. 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 [11], and root DNS servers [28]. The threat of floods has been used repeatedly by various criminal organizations to extort "protection money" from businesses with an online presence [24, 3]. 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 [20] 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. Further 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. The goal is to avoid congestion on the link 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. Collateral damage can occur for two reasons. A new source address is filtered out because it falls into an address range that is being actively filtered. Or, an address in the historic distribution of source address is filtered because the constraints we place on the number of ACL rules cause a portion of that distribution to be excluded. Our method of generateing ACL rules also allows for the possibility of accepting new legitimate sources that don't fit the historic distribution of source address.

We consider this a light-weight solution because no new hardware is required to implement the ACL rules, the computational requirements to derive the ACL rules are small, and a relatively small number of rules are required, 20 to 50 in our experiments. We consider any solution which requires little effort to implement, and allows a server to continue to function, significantly "raising the bar" against DDoS attacks. Trust issues are also minimal. Properly configured ACL rules only reference the destination address being protected therefore an ISP that allowed an end user to set the ACL rules need only confirm that the destination address and port belongs to that end user. Alternatively an ISP could offer this as a service. Filtering massive attacks is possible because the ACL rules performing the filtering can be installed at high speed routers.

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 [9] is a system that 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. The Teardrop, New Tear, Bonk, and Boink attacks use malformed IP fragments to crash or reboot vulnerable Windows machines. 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 [18]. 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 this 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. 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 [25] 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 [4] 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.

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 [27]. Tests that rely on detecting spoofing cannot 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 taking packets 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 solutions to the DDoS problem

Some of the proposed DDoS defenses take more radical steps to provide a definitive solution. Extending the Internet architecture with capabilities [29, 30] 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 [19, 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.

There are numerous commercial solutions implementing DDoS filtering at the victim or the victim's ISP [23, 22, 5, 7] (see Appendix B of [18] 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. Public material describes the architecture that allows these 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

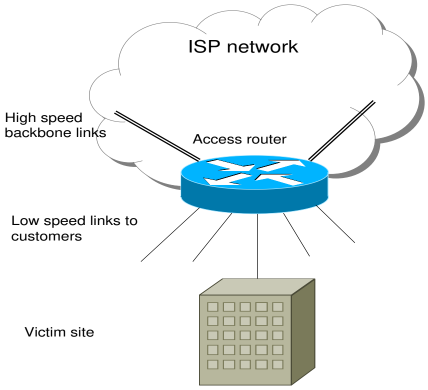

Figure 1: ACL rules can be applied at the access router or deeper in the ISP's network.

of these solutions do source IP based filtering as proposed in this paper, and we believe that these solutions could achieve better filtering by also employing some of the ideas presented here. 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 [26] uses a similar approach based on redirection to a high bandwidth data center that performs traffic cleaning using custom tools.

3 Source address prefix based filtering of DDoS floods

In this section we describe the test we propose for discarding enough packets to keep the link uncongested and yet allow the majority of legitimate traffic through. 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 on-going attack. The filtering algorithm generates a list of ACL rules for routers at the victim's ISP (for example the access router in Figure 1) that filter *the traffic sent to the victim* based on source IP addresses. The filtering algorithm ensures that the filtered traffic doesn't congest the bottleneck link (for example the victim's low speed link to the ISP in Figure 1) while the collateral damage is minimized and the number of rules kept within a prespecified 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 bottleneck link bandwidth 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 [6], 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 the destination IP address, protocol and destination port of the victim it protects. In our experiments we typically limit the number of rules to 100.

3.1 Evading source address prefix based filtering

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. But if the attacker can break into these systems, she can probably take the victim off-line without 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 Source address prefix based filtering as part of broader DDOS 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. It could be used by DDoS defenses in combination with other tests. SAPF could be used in combination with traffic scrubbing approaches when the attack is larger than the capacity of the available traffic scrubbing appliance: routers within the ISP could filter out some of the traffic directed at the victim, while the appliance would apply its filtering to the remaining traffic. Our current approaches filter out source prefixes that send much traffic during the attack, but not during normal operation. It is possible to combine our approach with other tests, for example to filter out source prefixes that have an unusually large percentage of incomplete connections. For attacks large enough to congest the backbone links of the access router connecting the victim (see Figure 1), SAPF could be applied at multiple routers, deeper in the network. In this paper we evaluate automatic algorithms for filtering out flood traffic, but we expect that in actual DDoS defenses network administrators would be able to override or modify the ACLs. For example they might forbid the algorithm from filtering out some important, low volume customers whom the algorithm could discriminate against. Network administrators might also assign weights to different prefixes of client addresses, to bias our algorithms towards protecting more important clients. 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 Discussion of algorithms generating ACL rules

In this section we briefly discuss the algorithms we propose for generating the ACL rules. Appendix A 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 some kind of 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 bottleneck link capacity. 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 of collateral damage

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 synthetic DDoS floods combined with actual NetFlow data on the legitimate traffic of various servers to evaluate the amount of collateral damage under 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 used

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)[1], and the largest DNS server (peak traffic 1.36 MB/5 minute bin). For the Campus data set we use the largest mail 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 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

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

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 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.

Table 1 and 9 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.

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.

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.

In summary, 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).

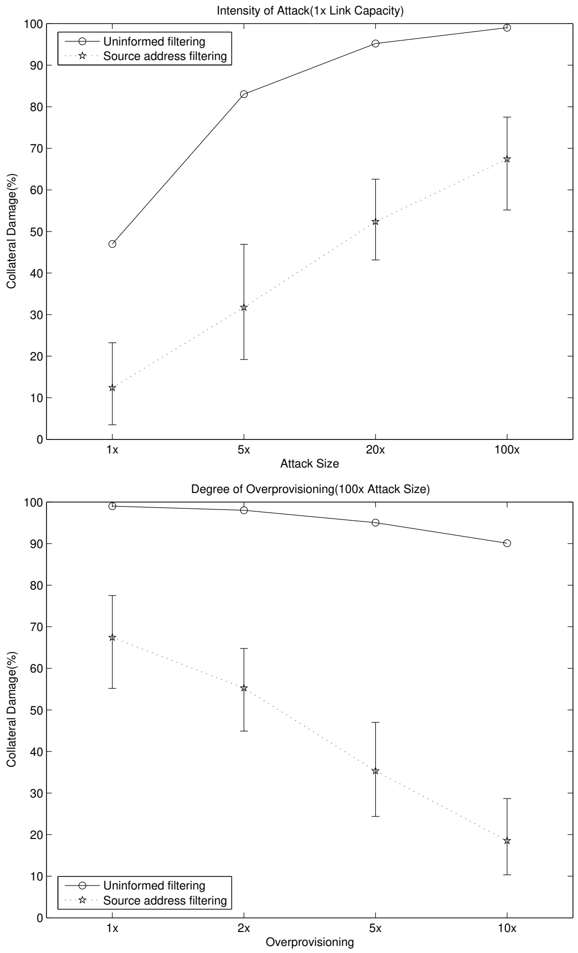

Figure 2: The effect of increased attack sizes and increased overprovisioning for the ISP web server.

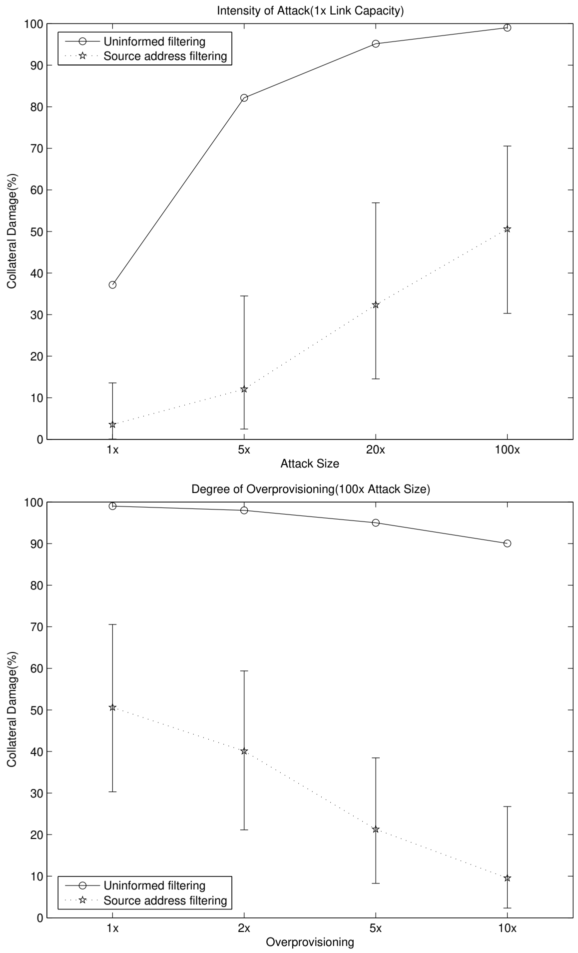

Figure 3: The effect of increased attack sizes and increased overprovisioning for the Campus mail server.

4.3 Intensity of attack and degree of overprovisioning

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 filtering

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%
Campus 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.

methods against very large attacks.

In summary, our most important conclusions are as follows.

- 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 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

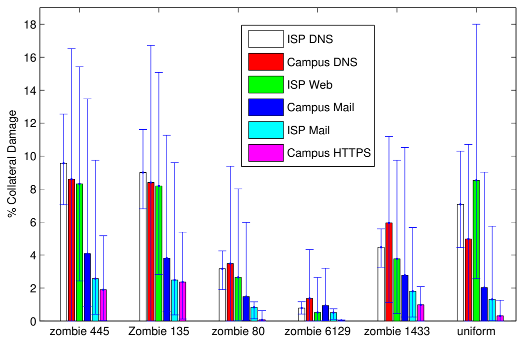

Figure 4: Attacks with different zombie networks using their own source addresses or spoofing uniformly at random.

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).

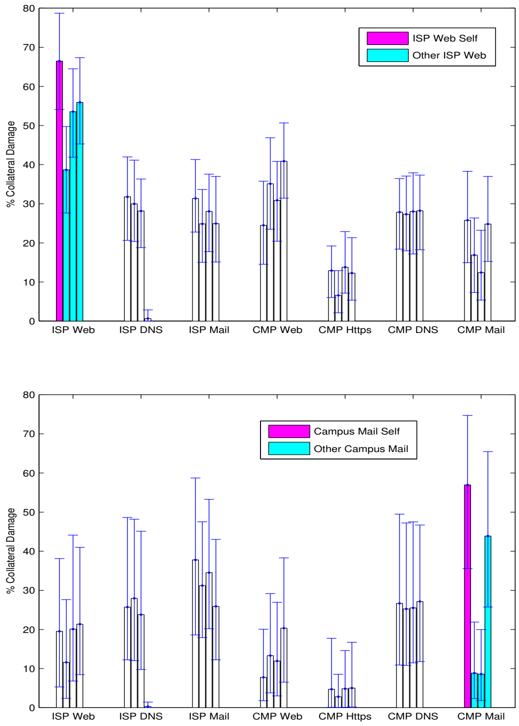

Figure 5: Attacks with source address distributions modeled after the traffic of servers of the same type as the victim can be very damaging.

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 one attacks against the largest Campus mail server. We also included the results for attacks modeled 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.

In summary, 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 is highly dependent on the choice of model server.

4.5 Number of ACL rules allowed

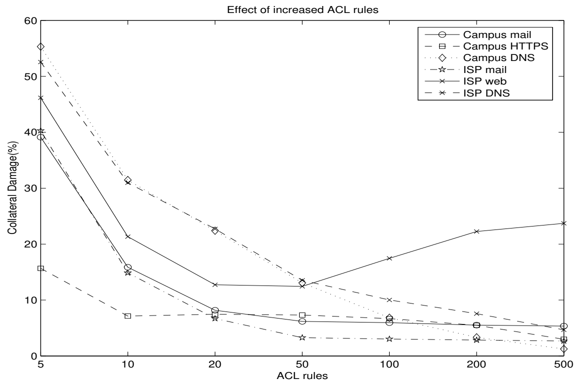

Figure 6: Typically collateral damage decreases as we increase the number of rules.

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.

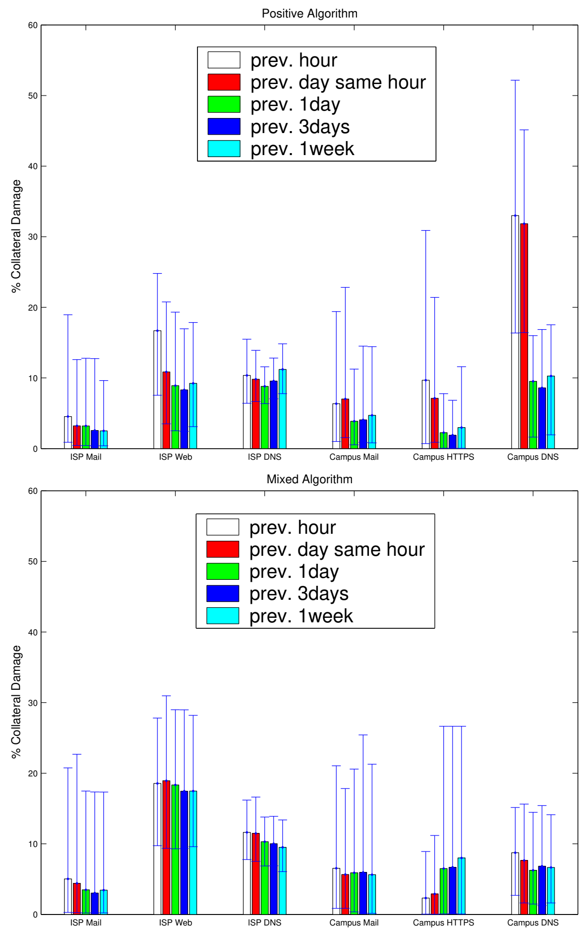

Figure 7: The effect of different baselines for the positive and mixed algorithm.

4.6 Choice of baseline and filtering algorithm

We experimented with using various portions of the traffic log of the victim as baselines for the 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. “Salting” the address space by low intensity probes by attackers is unlikely to have much impact on historic source distribution. If attackers manage to skew the source distribution with higer intensity probes a higher than expected amount of collateral damage might occur. Should this occur a switch to an older baseline could remedy the situation.

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. 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 approach to filtering attack traffic based on the historic distribution of packet source addresses arriving at a given IP address and service port. 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 allow the link to remain uncongested and allow a majority of the legitimate traffic to pass. We have also demonstrated a relative insensitivity to baseline choice. “Salting” the address space by low intensity probes by attackers is unlikely to have much impact on historic source distribution. If attackers manage to skew the source distribution with higer intensity probes a higher than expected amount of collateral damage might occur. Should this happen a switch to an older baseline could remedy the situation. We propose 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 the capacity of the bottleneck link. We use 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 link capacity, 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. From the experimental results we conclude that source address prefix based filtering can improve DDoS defenses. We see two specific settings in which it can be most useful: it can complement aproaches based on detecting packets with spoofed source addresses because it is most effective against attacks using the real addresses of the zombies, and it can act as an in-network prefilter for very large attacks that exceed the filtering capacity of available traffic scrubbing devices.

We consider this a light-weight solution because no new hardware is required to implement the ACL rules, the computational requirements to derive the ACL rules are small, and a small number of rules, 20 to 50 in our experiments, are required. We consider any solution which requires little effort to implement, and allows a server to continue to function, significantly "raising the bar" against DDoS attacks. Trust issues are also minimal. Properly configured ACL rules only reference the destination address being protected therefore an ISP that allowed an end user to set the ACL rules need only confirm that the destination address and port belongs to that end user. Alternatively an ISP could offer this as a service. Filtering massive attacks is possible because the ACL rules performing the filtering can be installed at high speed routers.

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 the flow records of ISP traffic.

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] Scott Berinato. How a bookmaker and a whiz kid took on an extortionist and won, May 2005.

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

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

[6] 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`.

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

[8] Cristian Estan, Stefan Savage, and George Varghese. Automatically inferring patterns of resource consumption in network traffic. In *Proceedings of the ACM SIGCOMM*, August 2003.

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

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

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

[12] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The elements of statistical learning. Springer, 2001. pages 453-479.

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

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

[20] 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.

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

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

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

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

[25] 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.

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

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

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

[29] 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.

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

A Description of algorithms for generating filtering rules

We describe three greedy algorithms for solving the following problem: given a set of flow records describing the current traffic directed to the server under attack and a set of flow records describing the historic "baseline" legitimate traffic, generate ACL rules to filter out the attack traffic. There are two important constraints on the solution: the total traffic passing the proposed ACL rules should not exceed the link capacity and the number of ACL rules must be within a given budget. Note that all filtering rules apply only to traffic sent to the victim, other traffic sharing the bottleneck link with the victim will not be affected. In practice, the "link capacity" argument we pass to the algorithms will not be the actual capacity of the bottleneck link, but a smaller number to accomodate other traffic and small variations in the traffic to the victim. Our algorithms are oblivious to these adjustments.

The filtering rules select traffic based on source prefixes. We have three algorithms corresponding to three different strategies. The first strategy is to drop traffic to the victim by default, and use ACL rules to allow traffic from certain source prefixes to pass through. The second strategy is to allow all traffic by default, but have ACL rules drop traffic from certain source prefixes. The third strategy is to use traffic cluster analysis [8] to find a set of prefixes with roughly the same current traffic and generate a mixed set of accept and deny rules allowing the prefixes with most legitimate traffic and rejecting those with most attack traffic. We call the algorithm implementing the first strategy "positive algorithm", the one for the second strategy "negative algorithm", and the third "mixed algorithm".

A.1 Positive algorithm

The positive algorithm tries to produce a list of non-overlapping prefixes that cover a large fraction of the historical baseline (thus resulting in little collateral damage), but not much of the current traffic (thus avoiding congestion). To narrow down this exploration we first cluster the historical traffic and pick a small number of clusters to focus on (the number of clusters we use is always at least twice the number of ACL rules we want). The next step starts with one rule allowing one IP address from each cluster and it extends the rules greedily until the link capacity is reached. The final step reduces greedily the number of rules (redistributing the excess bandwidth resulting from the deleted rules) until the number of ACL rules fits within the budget.

Input: Number of ACL rules allowed $NumRules$ and link capacity $Capacity$.
Output: A list of prefixes whose current traffic is within the link capacity.
begin
  $Centroids \leftarrow \texttt{Clustering}(NumRules * 2)$
  $Room \leftarrow Capacity - \texttt{CrtTraffic}(Centroids)$
  $L \leftarrow \texttt{GenerateRules}(Centroids, Room)$
  $Room \leftarrow Capacity - \texttt{CrtTraffic}(L)$
  return $\texttt{ReduceRules}(L, NumRules, Room)$
end

Algorithm 1: The positive algorithm.

A.1.1 Clustering

We model historical traffic using Hierarchical Aglomerative Clustering(HAC)[12]. construct baseline models that will be combined with 2 graph are defined on prefixes and prefixes define a non-linear space. Since we are filtering based on prefixes it is natural to use a metric based on prefixes as the distance metric used in clustering. We use the following method to compute the distance between 2 prefixes: XOR the prefixes together, keep the highest order bit set and set the lower bits to 0. The resulting binary number is the distance between the prefixes. The distance metric multiplied by the sum of the traffic of the two prefixes produces the clustering metric that is applied in a single link HAC algorithm. We take cuts in the resulting dendrograms by clustering until a pre-determined number N of clusters are produced. We then define centroids as the longest path from the "top" of each sub dendrogram to the leaves. Ties in path length are broken by choosing the leaf nodes that were clustered first.

A.1.2 Creating the list of rules

The second step of the positive algorithm starts with a list of the centroids (individual IP addresses each representing a cluster) for the historical traffic and it greedily generalizes them reducing the prefix length until we obtain a list of non-overlapping prefixes whose traffic is barely below the link capacity. Emitting an allow ACL rule for each node in the list does not overflow the link capacity, but the number of rules might still be above our budget.

Positive starts with the centroids. In the rare case that the centroids allow too much traffic themselves we remove the worse centroid (highest current to historical ratio) until generating accept rules for all the remaining centroids does not go over the link capacity. The greedy rule for deciding which prefix to shorten is to pick the move for which the ratio between the newly covered historical traffic and the newly covered current traffic is maximized. This prefix shortening can result in reducing the number of rules as the shorter prefix might include more than one of the prefixes in the list.

Input: A list of prefixes L and the amount of extra capacity $Room$ on the link.
Output: A list of non-overlapping, more general prefixes that cover more historic traffic and whose current traffic is within the link capacity.
begin
  while $B \leftarrow \texttt{GetBestMove}(L, Room)$ **do**
    $L \leftarrow \texttt{RemPrefixes}(L, B.newprefix)$
    $L \leftarrow \texttt{Append}(L, B.newprefix)$
    $Room \leftarrow Room - B.DeltaC$
    $L \leftarrow \texttt{CombineSiblings}(L)$
  return L
end

Function `GenerateRules(`*L,Room*`)` shortens the prefixes in L to allow more traffic through.

```
Input: A list of prefixes L and the amount of extra capacity
       Room on the link.
Output: The prefix B that generalizes one of the prefixes in
        L so that the extra addresses it covers have the
        largest ratio between historic traffic and current
        traffic.
begin
    B ← NIL
    foreach N ∈ L do
        for M ← Parent(N) to 0.0.0.0/0 do
            DeltaC ← CrtTraf(M)
            DeltaH ← HistTraf(M)
            foreach E ∈ L do
                if IsDescendant(M, E) then
                    DeltaC ← DeltaC − CrtTraf(E)
                    DeltaH ←
                    DeltaH − HistTraf(E)
        if DeltaC ≤ Room and (B = NIL or
        DeltaH/DeltaC ≥ B.DeltaH/B.DeltaC)
        then
            B.newprefix ← M
            B.DeltaC ← DeltaC
            B.DeltaH ← DeltaH
    return B
end
```

Function `GetBestMove(L,Room)` makes a greedy choice on which prefix from L to shorten.

```
Input: A list of prefixes L and a prefix N.
Output: The list of prefixes in L which are not more
        specific than N.
begin
    foreach M ∈ L do
        if IsDescendant(N, M) then
            L = Remove(L, M)
    return L
end
```

Function `RemovePrefixes(L,N)` Removes all prefixes included in N from prefix list L

A.1.3 Shortening the list of rules

The third step of the positive algorithm reduces the length of the list of rules to within the ACL budget without overflowing the link capacity. The greedy decision criterion for removing a prefix is to pick the prefix with the worst (largest) ratio between the current and historical traffic it covers. Removing this prefix frees up as much link capacity as the amount of current traffic the prefix allowed. To consume the extra room (and possibly admit more good traffic, thereby decreasing collateral damage) we greedily take the best move.

A.2 Negative algorithm

The negative algorithm is the converse of the positive algorithm. The strategy is to allows all traffic by default and generate rules that deny traffic for certain prefixes. Thus the negative algorithm must find prefixes covering much of the current traffic but not much of the historical traffic. For the negative algorithm we perform the clustering on the current traffic at the time of the attack. The second step starts with a list of prefixes, each capturing an entire cluster, and it greedily increases the length of prefixes as long as the traffic captured by all rules is still at least as large as the amount of traffic we must filter out. In the last step the algorithm greedily combines pairs of prefixes from this list until the number of rules fits within the budget. Since at each step the combined prefix filters out more traffic than the prefixes it replaces, we also greedily increase the length of prefixes. The implementation details of this algorithm are different from those for the positive algorithm, but since their strategies are similar we omit the details for brevity.

A.3 Mixed algorithm

The mixed algorithm uses traffic clustering [8] to generate a list of overlapping source prefixes. Let $MatchingTraf(N)$ be the amount of current traffic for which prefix N is the most specific matching prefix. The traffic clustering algorithm takes a single parameter, a threshold T, and it has the property that for all prefixes in its result, $MatchingTraf(N)$ is above a T but below $2T$ (except if we have individual IP addresses with traffic larger than $2T$ which will be in the result). Thus if the ACL rules are ordered such that the more specific prefixes come first, each prefix controls a roughly equal share of the current traffic. The second step of the mixed algorithm decides which prefixes to deny and which to allow. We simply sort the list of prefixes by the ratio between the current and historic traffic they match in ascending order. We set prefixes from the head of the list to allow until we fill the link capacity and the rest of the prefixes we set to deny. The third step does not change the filtering at all but it attempts to reduce the number of rules used to express the filtering decision. This reduction step relies on two observations: if two prefixes are siblings (they differ only in the last bit) and they have the same action, they can be replaced by their parent performing the same action; and if a prefix has the same action as the most specific ancestor that includes it, the prefix can be removed from the list. The threshold T controls the number of clusters to some extent, but it is hard to predict how much the number of rules is reduced in the last step. We repeat a few runs of the mixed algorithm with progressively lower thresholds until we get a a number of rules close to our budget of rules.

B Detailed experimental results

This appendix presents more detailed measurement results for experiments discussed in section 4.

Model for source addresses in flood	Victim Server		
	Campus Mail	Campus HTTPS	Campus DNS
Campus Victim Mail	35.6/57.0/76.1%	0.6/3.5/8.7%	20.9/32.9/46.9%
Campus Mail2	2.4/8.8/22.4%	0.2/3.2/9.8%	7.5/17.5/26.5%
Campus Mail3	1.8/8.6/21.4%	0.1/2.0/8.0%	10.4/18.8/28.9%
Campus Mail4	25.8/43.9/68.2%	0.8/4.4/13.3%	16.0/27.9/43.1%
Campus Web1	1.8/7.8/21.4%	0.5/4.2/12.3%	13.3/21.2/33.4%
Campus Web2	3.8/13.3/29.2%	5.7/12.3/26.9%	9.1/21.1/35.1%
Campus Web3	3.0/11.9/29.0%	0.9/4.2/12.9%	13.7/23.3/33.2%
Campus Web4	6.5/20.3/39.4%	2.8/9.2/20.1%	10.3/21.1/32.2%
Campus Victim HTTPS	0.1/4.7/19.7%	22.8/36.9/58.0%	0.0/6.2/12.9%
Campus HTTPS2	0.1/2.8/8.9%	11.8/23.2/37.4%	0.0/4.4/9.7%
Campus HTTPS3	0.1/4.9/17.8%	16.6/31.2/52.4%	0.0/6.4/14.7%
Campus HTTPS4	0.2/5.0/17.8%	19.6/30.4/53.2%	0.0/5.6/11.7%
Campus Victim DNS	10.9/26.7/55.0%	0.7/3.8/10.4%	47.1/60.8/72.2%
Campus DNS2	10.8/25.2/50.8%	0.5/3.7/10.4%	37.0/51.5/65.2%
Campus DNS3	11.5/25.5/52.0%	0.7/3.9/11.9%	37.1/52.1/66.6%
Campus DNS4	11.8/27.1/50.6%	0.7/3.9/10.6%	39.8/51.7/65.0%
ISP Mail1	18.6/37.8/58.7%	0.1/2.8/5.4%	16.4/28.9/40.5%
ISP Mail2	17.9/31.2/51.7%	0.1/2.0/4.9%	15.0/26.6/38.8%
ISP Mail3	20.2/34.5/54.0%	0.2/2.3/4.7%	18.9/30.6/45.6%
ISP Mail4	12.3/25.9/45.0%	0.0/1.7/4.6%	12.4/23.2/36.2%
ISP Web1	5.3/19.5/39.6%	1.3/4.5/11.1%	15.8/25.1/35.5%
ISP Web2	2.4/11.6/34.4%	0.2/3.3/9.8%	6.6/14.7/25.1%
ISP Web3	6.8/20.1/44.3%	1.2/4.5/10.2%	11.7/22.1/31.3%
ISP Web4	8.5/21.4/41.2%	1.3/5.0/15.4%	13.6/25.2/35.8%
ISP DNS1	12.2/25.7/50.4%	0.3/3.5/12.0%	27.5/41.8/56.0%
ISP DNS2	12.0/28.0/51.4%	0.3/3.5/10.6%	27.1/41.9/53.6%
ISP DNS3	9.8/23.8/45.8%	0.1/2.6/7.2%	28.7/40.2/53.3%
ISP DNS4	0.0/0.3/1.5%	0.0/0.1/0.5%	0.0/0.6/2.7%

Table 5: Attacks against Campus victim servers are modeled after the distributions of top 4 servers of each service in Campus and ISP.

Model for source addresses in flood	Victim Server		
	ISP Mail	ISP Web	ISP DNS
Campus Mail1	13.6/30.5/52.1%	14.9/25.8/38.5%	33.6/38.3/41.8%
Campus Mail2	1.6/6.3/20.5%	7.3/16.9/27.9%	15.0/17.8/21.6%
Campus Mail3	0.8/4.4/15.0%	5.4/12.4/23.2%	15.4/18.6/22.3%
Campus Mail4	11.7/29.6/49.2%	15.3/24.8/38.3%	30.7/34.5/38.6%
Campus Web1	1.0/7.1/25.6%	14.5/24.5/36.6%	15.0/18.2/21.5%
Campus Web2	1.9/9.2/26.9%	23.5/35.1/48.4%	21.1/25.0/29.2%
Campus Web3	1.1/7.3/24.5%	20.4/30.9/40.9%	19.8/23.4/27.9%
Campus Web4	3.7/14.1/34.9%	31.4/40.9/50.7%	22.8/26.9/30.5%
Campus HTTPS1	0.3/3.8/13.5%	6.0/12.9/20.2%	7.0/9.9/13.4%
Campus HTTPS2	0.2/3.1/11.8%	2.1/6.6/13.4%	3.1/4.7/6.9%
Campus HTTPS3	0.3/4.0/13.9%	7.2/13.8/24.4%	8.1/11.0/14.6%
Campus HTTPS4	0.3/4.1/15.9%	5.4/12.3/21.4%	5.1/8.0/11.2%
Campus DNS1	7.5/20.1/39.6%	18.4/27.9/36.5%	40.3/44.5/48.8%
Campus DNS2	6.6/19.2/40.2%	18.0/27.3/38.1%	38.7/42.9/46.8%
Campus DNS3	5.7/18.6/37.4%	17.2/28.0/38.0%	37.2/41.6/45.4%
Campus DNS4	6.0/18.4/37.4%	18.2/28.2/39.0%	43.1/47.1/51.1%
ISP Victim Mail	39.4/56.8/75.4%	22.8/31.4/41.9%	36.8/40.5/44.7%
ISP Mail2	16.9/38.2/62.6%	15.1/24.9/34.7%	34.9/39.7/45.8%
ISP Mail3	22.3/43.9/69.7%	17.8/28.0/37.9%	35.3/39.2/42.5%
ISP Mail4	16.2/30.9/49.3%	15.1/24.9/37.0%	26.7/31.2/34.9%
ISP Victim Web	5.2/18.2/42.1%	54.1/66.5/79.4%	27.8/31.5/35.4%
ISP Web2	2.0/11.4/30.7%	27.6/38.7/50.1%	20.3/24.3/27.9%
ISP Web3	4.0/16.8/40.7%	41.9/53.5/65.2%	28.1/31.9/36.2%
ISP Web4	7.0/19.4/42.8%	45.3/55.9/67.3%	30.9/34.8/38.5%
ISP DNS1	8.7/22.5/45.0%	20.4/30.0/41.4%	56.6/60.5/64.3%
ISP Victim DNS	9.2/23.9/46.6%	20.6/31.8/42.0%	61.9/66.1/69.5%
ISP DNS3	8.0/20.1/39.0%	18.8/28.1/36.4%	49.8/53.2/56.7%
ISP DNS4	0.0/0.4/1.5%	0.0/0.7/3.0%	0.5/1.0/1.8%

Table 6: Attacks against ISP victim servers are modeled after the distributions of top 4 servers of each service in Campus and ISP.

Victim server	Algorithm		
	Positive	Mixed	Negative
ISP Mail	0.4 / 2.6 / 12.8%	0.2 / 3.0 / 17.4%	1.2 / 15.1 / 35.6%
ISP Web	2.4 / 8.3 / 17.0%	9.3 / 17.5 / 29.0%	4.6 / 12.6 / 22.9%
ISP DNS	7.0 / 9.6 / 12.8%	6.9 / 10.0 / 13.9%	22.1 / 36.2 / 48.4%
Campus Mail	0.9 / 4.1 / 14.5%	0.2 / 6.0 / 25.4%	3.7 / 16.3 / 36.8%
Campus HTTPS	0.0 / 1.9 / 6.8%	0.1 / 6.7 / 26.6%	2.3 / 7.2 / 23.8%
Campus DNS	0.0 / 8.6 / 16.9%	1.2 / 6.8 / 15.4%	16.1 / 33.0 / 57.8%

Table 7: The collateral damage with positive, mixed, and negative algorithm for default parameters.

Victim server	Attack Sizes	Overprovisioning			
		1x	2x	5x	10x
ISP Mail	5x	1.6 / 8.0 / 22.8%	0.4 / 2.6 / 12.8%	0.0 / 0.0 / 0.1%	0.0 / 0.0 / 0.0%
	20x	13.5 / 31.7 / 51.6%	2.6 / 12.1 / 29.8%	0.6 / 3.0 / 13.1%	0.2 / 0.9 / 1.5%
	100x	32.4 / 54.4 / 76.1%	20.9 / 43.3 / 60.2%	9.8 / 23.8 / 42.7%	1.7 / 8.3 / 24.2%
ISP Web	5x	19.2 / 31.8 / 46.9%	2.4 / 8.3 / 17.0%	0.0 / 0.4 / 2.9%	0.0 / 0.0 / 0.0%
	20x	43.1 / 52.4 / 62.6%	18.7 / 29.8 / 41.3%	2.5 / 7.2 / 13.0%	0.0 / 1.8 / 5.2%
	100x	55.2 / 67.4 / 77.5%	45.0 / 55.3 / 64.8%	24.4 / 35.3 / 47.0%	10.3 / 18.5 / 28.7%
ISP DNS	5x	29.5 / 35.8 / 40.9%	7.1 / 9.6 / 12.8%	0.0 / 0.1 / 0.3%	0.0 / 0.0 / 0.0%
	20x	45.5 / 49.7 / 54.6%	27.3 / 30.6 / 34.1%	5.6 / 7.0 / 8.3%	0.7 / 1.1 / 1.6%
	100x	60.1 / 64.5 / 67.5%	47.5 / 51.2 / 55.3%	27.4 / 31.1 / 35.2%	14.8 / 17.1 / 20.1%
Campus Mail	5x	2.5 / 12.1 / 34.5%	0.9 / 4.1 / 14.5%	0.0 / 0.0 / 0.3%	0.0 / 0.0 / 0.0%
	20x	14.5 / 32.4 / 56.9%	4.6 / 14.0 / 29.0%	1.0 / 4.0 / 11.2%	0.1 / 1.2 / 3.8%
	100x	30.3 / 50.6 / 70.6%	21.2 / 40.1 / 59.4%	8.3 / 21.3 / 38.5%	2.3 / 9.6 / 26.7%
Campus HTTPS	5x	1.2 / 5.6 / 18.1%	0.0 / 1.9 / 6.8%	0.0 / 0.0 / 0.0%	0.0 / 0.0 / 0.0%
	20x	2.8 / 8.5 / 18.1%	1.6 / 5.8 / 13.5%	0.1 / 2.2 / 5.4%	0.0 / 0.4 / 1.4%
	100x	4.7 / 11.2 / 21.2%	3.2 / 9.3 / 18.7%	2.1 / 5.9 / 13.9%	1.4 / 4.2 / 9.0%
Campus DNS	5x	15.3 / 26.3 / 38.0%	0.0 / 8.6 / 16.9%	0.0 / 0.2 / 0.0%	0.0 / 0.0 / 0.0%
	20x	34.9 / 49.0 / 63.5%	22.0 / 34.7 / 46.3%	2.6 / 9.0 / 16.9%	0.0 / 2.1 / 8.5%
	100x	55.0 / 69.2 / 78.6%	43.4 / 59.3 / 73.4%	29.7 / 39.7 / 51.5%	14.7 / 25.8 / 36.3%

Table 8: The effect of attack size and overprovisioning on the collateral damage of SAPF.

Victim server	Algorithm		
	Positive	Mixed	Negative
ISP Mail	0.4 / 2.6 / 12.8%	0.2 / 3.0 / 17.4%	1.2 / 15.1 / 35.6%
ISP Web	2.4 / 8.3 / 17.0%	9.3 / 17.5 / 29.0%	4.6 / 12.6 / 22.9%
ISP DNS	7.0 / 9.6 / 12.8%	6.9 / 10.0 / 13.9%	22.1 / 36.2 / 48.4%
Campus Mail	0.9 / 4.1 / 14.5%	0.2 / 6.0 / 25.4%	3.7 / 16.3 / 36.8%
Campus HTTPS	0.0 / 1.9 / 6.8%	0.1 / 6.7 / 26.6%	2.3 / 7.2 / 23.8%
Campus DNS	0.0 / 8.6 / 16.9%	1.2 / 6.8 / 15.4%	16.1 / 33.0 / 57.8%

Table 9: The collateral damage with positive, mixed, and negative algorithm for default parameters.

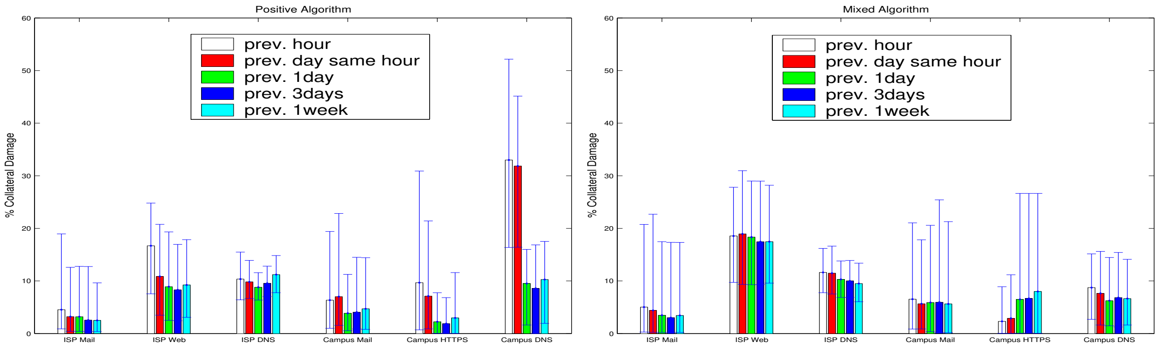

Figure 8: The effect of different baselines for the positive and mixed algorithm.

Victim server	Attack Sizes	Overprovisioning			
		1x	2x	5x	10x
ISP Mail	5x	1.6 / 8.0 / 22.8%	0.4 / 2.6 / 12.8%	0.0 / 0.0 / 0.1%	0.0 / 0.0 / 0.0%
	20x	13.5 / 31.7 / 51.6%	2.6 / 12.1 / 29.8%	0.6 / 3.0 / 13.1%	0.2 / 0.9 / 1.5%
	100x	32.4 / 54.4 / 76.1%	20.9 / 43.3 / 60.2%	9.8 / 23.8 / 42.7%	1.7 / 8.3 / 24.2%
ISP Web	5x	19.2 / 31.8 / 46.9%	2.4 / 8.3 / 17.0%	0.0 / 0.4 / 2.9%	0.0 / 0.0 / 0.0%
	20x	43.1 / 52.4 / 62.6%	18.7 / 29.8 / 41.3%	2.5 / 7.2 / 13.0%	0.0 / 1.8 / 5.2%
	100x	55.2 / 67.4 / 77.5%	45.0 / 55.3 / 64.8%	24.4 / 35.3 / 47.0%	10.3 / 18.5 / 28.7%
ISP DNS	5x	29.5 / 35.8 / 40.9%	7.1 / 9.6 / 12.8%	0.0 / 0.1 / 0.3%	0.0 / 0.0 / 0.0%
	20x	45.5 / 49.7 / 54.6%	27.3 / 30.6 / 34.1%	5.6 / 7.0 / 8.3%	0.7 / 1.1 / 1.6%
	100x	60.1 / 64.5 / 67.5%	47.5 / 51.2 / 55.3%	27.4 / 31.1 / 35.2%	14.8 / 17.1 / 20.1%
Campus Mail	5x	2.5 / 12.1 / 34.5%	0.9 / 4.1 / 14.5%	0.0 / 0.0 / 0.3%	0.0 / 0.0 / 0.0%
	20x	14.5 / 32.4 / 56.9%	4.6 / 14.0 / 29.0%	1.0 / 4.0 / 11.2%	0.1 / 1.2 / 3.8%
	100x	30.3 / 50.6 / 70.6%	21.2 / 40.1 / 59.4%	8.3 / 21.3 / 38.5%	2.3 / 9.6 / 26.7%
Campus HTTPS	5x	1.2 / 5.6 / 18.1%	0.0 / 1.9 / 6.8%	0.0 / 0.0 / 0.0%	0.0 / 0.0 / 0.0%
	20x	2.8 / 8.5 / 18.1%	1.6 / 5.8 / 13.5%	0.1 / 2.2 / 5.4%	0.0 / 0.4 / 1.4%
	100x	4.7 / 11.2 / 21.2%	3.2 / 9.3 / 18.7%	2.1 / 5.9 / 13.9%	1.4 / 4.2 / 9.0%
Campus DNS	5x	15.3 / 26.3 / 38.0%	0.0 / 8.6 / 16.9%	0.0 / 0.2 / 0.0%	0.0 / 0.0 / 0.0%
	20x	34.9 / 49.0 / 63.5%	22.0 / 34.7 / 46.3%	2.6 / 9.0 / 16.9%	0.0 / 2.1 / 8.5%
	100x	55.0 / 69.2 / 78.6%	43.4 / 59.3 / 73.4%	29.7 / 39.7 / 51.5%	14.7 / 25.8 / 36.3%

Table 10: The effect of attack size and overprovisioning on collateral damage.

Model for source addresses in flood	Victim Server		
	Campus Mail	Campus HTTPS	Campus DNS
Campus Victim Mail	35.6/57.0/76.1%	0.6/3.5/8.7%	20.9/32.9/46.9%
Campus Mail2	2.4/8.8/22.4%	0.2/3.2/9.8%	7.5/17.5/26.5%
Campus Mail3	1.8/8.6/21.4%	0.1/2.0/8.0%	10.4/18.8/28.9%
Campus Mail4	25.8/43.9/68.2%	0.8/4.4/13.3%	16.0/27.9/43.1%
Campus Web1	1.8/7.8/21.4%	0.5/4.2/12.3%	13.3/21.2/33.4%
Campus Web2	3.8/13.3/29.2%	5.7/12.3/26.9%	9.1/21.1/35.1%
Campus Web3	3.0/11.9/29.0%	0.9/4.2/12.9%	13.7/23.3/33.2%
Campus Web4	6.5/20.3/39.4%	2.8/9.2/20.1%	10.3/21.1/32.2%
Campus Victim HTTPS	0.1/4.7/19.7%	22.8/36.9/58.0%	0.0/6.2/12.9%
Campus HTTPS2	0.1/2.8/8.9%	11.8/23.2/37.4%	0.0/4.4/9.7%
Campus HTTPS3	0.1/4.9/17.8%	16.6/31.2/52.4%	0.0/6.4/14.7%
Campus HTTPS4	0.2/5.0/17.8%	19.6/30.4/53.2%	0.0/5.6/11.7%
Campus Victim DNS	10.9/26.7/55.0%	0.7/3.8/10.4%	47.1/60.8/72.2%
Campus DNS2	10.8/25.2/50.8%	0.5/3.7/10.4%	37.0/51.5/65.2%
Campus DNS3	11.5/25.5/52.0%	0.7/3.9/11.9%	37.1/52.1/66.6%
Campus DNS4	11.8/27.1/50.6%	0.7/3.9/10.6%	39.8/51.7/65.0%
ISP Mail1	18.6/37.8/58.7%	0.1/2.8/5.4%	16.4/28.9/40.5%
ISP Mail2	17.9/31.2/51.7%	0.1/2.0/4.9%	15.0/26.6/38.8%
ISP Mail3	20.2/34.5/54.0%	0.2/2.3/4.7%	18.9/30.6/45.6%
ISP Mail4	12.3/25.9/45.0%	0.0/1.7/4.6%	12.4/23.2/36.2%
ISP Web1	5.3/19.5/39.6%	1.3/4.5/11.1%	15.8/25.1/35.5%
ISP Web2	2.4/11.6/34.4%	0.2/3.3/9.8%	6.6/14.7/25.1%
ISP Web3	6.8/20.1/44.3%	1.2/4.5/10.2%	11.7/22.1/31.3%
ISP Web4	8.5/21.4/41.2%	1.3/5.0/15.4%	13.6/25.2/35.8%
ISP DNS1	12.2/25.7/50.4%	0.3/3.5/12.0%	27.5/41.8/56.0%
ISP DNS2	12.0/28.0/51.4%	0.3/3.5/10.6%	27.1/41.9/53.6%
ISP DNS3	9.8/23.8/45.8%	0.1/2.6/7.2%	28.7/40.2/53.3%
ISP DNS4	0.0/0.3/1.5%	0.0/0.1/0.5%	0.0/0.6/2.7%

Table 11: Attacks against Campus victim servers are modeled after the distributions of top 4 servers of each service in Campus and ISP.

Model for source addresses in flood	Victim Server		
	ISP Mail	ISP Web	ISP DNS
Campus Mail1	13.6/30.5/52.1%	14.9/25.8/38.5%	33.6/38.3/41.8%
Campus Mail2	1.6/6.3/20.5%	7.3/16.9/27.9%	15.0/17.8/21.6%
Campus Mail3	0.8/4.4/15.0%	5.4/12.4/23.2%	15.4/18.6/22.3%
Campus Mail4	11.7/29.6/49.2%	15.3/24.8/38.3%	30.7/34.5/38.6%
Campus Web1	1.0/7.1/25.6%	14.5/24.5/36.6%	15.0/18.2/21.5%
Campus Web2	1.9/9.2/26.9%	23.5/35.1/48.4%	21.1/25.0/29.2%
Campus Web3	1.1/7.3/24.5%	20.4/30.9/40.9%	19.8/23.4/27.9%
Campus Web4	3.7/14.1/34.9%	31.4/40.9/50.7%	22.8/26.9/30.5%
Campus HTTPS1	0.3/3.8/13.5%	6.0/12.9/20.2%	7.0/9.9/13.4%
Campus HTTPS2	0.2/3.1/11.8%	2.1/6.6/13.4%	3.1/4.7/6.9%
Campus HTTPS3	0.3/4.0/13.9%	7.2/13.8/24.4%	8.1/11.0/14.6%
Campus HTTPS4	0.3/4.1/15.9%	5.4/12.3/21.4%	5.1/8.0/11.2%
Campus DNS1	7.5/20.1/39.6%	18.4/27.9/36.5%	40.3/44.5/48.8%
Campus DNS2	6.6/19.2/40.2%	18.0/27.3/38.1%	38.7/42.9/46.8%
Campus DNS3	5.7/18.6/37.4%	17.2/28.0/38.0%	37.2/41.6/45.4%
Campus DNS4	6.0/18.4/37.4%	18.2/28.2/39.0%	43.1/47.1/51.1%
ISP Victim Mail	39.4/56.8/75.4%	22.8/31.4/41.9%	36.8/40.5/44.7%
ISP Mail2	16.9/38.2/62.6%	15.1/24.9/34.7%	34.9/39.7/45.8%
ISP Mail3	22.3/43.9/69.7%	17.8/28.0/37.9%	35.3/39.2/42.5%
ISP Mail4	16.2/30.9/49.3%	15.1/24.9/37.0%	26.7/31.2/34.9%
ISP Victim Web	5.2/18.2/42.1%	54.1/66.5/79.4%	27.8/31.5/35.4%
ISP Web2	2.0/11.4/30.7%	27.6/38.7/50.1%	20.3/24.3/27.9%
ISP Web3	4.0/16.8/40.7%	41.9/53.5/65.2%	28.1/31.9/36.2%
ISP Web4	7.0/19.4/42.8%	45.3/55.9/67.3%	30.9/34.8/38.5%
ISP DNS1	8.7/22.5/45.0%	20.4/30.0/41.4%	56.6/60.5/64.3%
ISP Victim DNS	9.2/23.9/46.6%	20.6/31.8/42.0%	61.9/66.1/69.5%
ISP DNS3	8.0/20.1/39.0%	18.8/28.1/36.4%	49.8/53.2/56.7%
ISP DNS4	0.0/0.4/1.5%	0.0/0.7/3.0%	0.5/1.0/1.8%

Table 12: Attacks against ISP victim servers are modeled after the distributions of top 4 servers of each service in Campus and ISP.