PRIMED: Community-of-Interest-Based DDoS Mitigation

Patrick Verkaik, Oliver Spatscheck†, Jacobus Van der Merwe†, and Alex C. Snoeren
University of California, San Diego / †AT&T Labs-Research

ABSTRACT

Most existing distributed denial-of-service (DDoS) mitigation proposals are reactive in nature, *i.e.*, they are deployed to limit the damage caused by attacks after they are detected. In contrast, we present PRIMED, a proactive approach to DDoS mitigation that allows users to specify to their ISP *a priori* their (dis)interest in receiving traffic from particular network entities. Our solution employs *communities of interest* (COIs) to capture the collective past behavior of remote network entities and uses them to predict future behavior. Specifically, ISPs construct a network-wide *bad COI* that contains network entities who exhibited unwanted behavior in the past, and per-customer *good COIs* containing remote network entities that have previously engaged in legitimate communication with the customer. Our system uses these derived sets together with customer-specific policies to proactively mitigate DDoS attacks using existing router mechanisms. Indeed, preliminary lab testing shows that our approach is deployable on modern edge router platforms without degrading packet forwarding performance. This implies that our approach offers DDoS protection at a truly massive scale, *i.e.*, every customer access link. Simulation results show that our approach improves protection against 91–93% of actual DDoS attacks on real customers—providing complete protection against 38–53% of such attacks—while slightly increasing vulnerability in only 5–7% of attacks.

Categories and Subject Descriptors

C.2.6 [**Computer-Communication Networks**]: Internetworking

General Terms

Design, Security

Keywords

Denial of Service,Communities of Interest

1. INTRODUCTION

The Internet's any-to-any, best-effort communication model is widely heralded as one of the main reasons for its success. The fact that the network will try its best to deliver packets to the destination identified in each packet header, however, is also the source of a large and growing problem on the Internet, namely denial-of-service (DoS) and distributed denial-of-service (DDoS) attacks. Restricting or controlling the any-to-any nature of IP communication is at the root of many recently proposed DDoS mitigation techniques [7, 11, 25]. Most existing solutions, however, are reactive in nature: mitigation is only triggered once an attack has been detected. With the advent of large-scale botnets, attackers can launch massive flooding attacks that start almost instantaneously, effectively knocking hosts or even entire networks off the Internet before they have a chance to react [19]. Hence, we advocate proactive, predictive mechanisms that are ready to defend against DDoS attacks before they happen. Furthermore, we suggest that these proactive techniques can be effectively and efficiently implemented by network service providers. By providing a broad first line of defense, proactive mechanisms can buy time for more focused, reactive mechanisms to be activated.

The key challenge that must be overcome to successfully deploy proactive mitigation is to determine what traffic is likely to be malignant without having the opportunity to inspect it. Our solution follows from the basic observation that past behavior is often a good predictor of future behavior. Specifically, rather than focus on traffic content—a potentially complex and resource-intensive process—we instead consider the history of the communicating parties. We presume that participation of a network entity (*e.g.*, an IP address, subnet, autonomous system, or router ingress interface) in malicious behavior against a host or network may be an indication of potential future unwanted behavior against the same or other victims. Similarly, we use past legitimate communication between network entities as an indicator of potential future legitimate communication between the particular parties. In terms of collecting communication histories, we observe that ISPs are well positioned to monitor and record malicious behavior across their entire network, while individual end hosts or stub networks are in the best position to determine which traffic they receive is actually desirable.

In keeping with recent work we use the term *communities of interest (COIs)* to refer to the set of communicating entities [3, 8, 14]. Thus a *bad COI* is a set of network entities that previously engaged in unwanted behavior, whereas a *good COI* contains network entities that participated in legitimate communication with a particular destination (host or stub network). We combine these good and bad COIs with destination-specific policies to proactively restrict any-to-any communication in an attempt to predictively mitigate against upcoming DDoS attacks.

In this paper, we focus on the deployment of proactive DDoS mitigation by an Internet service provider on behalf of its customers. In particular, we allow a provider to protect customer traffic *up until the customer's access link*, after which the customer has the

SIGCOMM'06 Workshops September 11-15, 2006, Pisa, Italy.

ability to protect its own infrastructure and servers. In that context, we make two main contributions: First, we present the PRIMED architecture, which allows customers of an ISP to control their any-to-any connectivity in an informed manner by balancing the utility of communication against the risk of reachability. Second, we evaluate the feasibility of our approach through a case study of traces from a number of commercial Web servers and an independently collected trace of DDoS attacks from a tier-one ISP.

The remainder of this paper is organized as follows. We begin in Section 2 with a brief survey of related work. Section 3 describes our PRIMED architecture. In Section 4 we describe our experimental setup and the data sources used for our evaluation. In Section 5 we examine the properties of the derived communities of interest and in Section 6 we present an evaluation of the effectiveness of our approach. We conclude by discussing the broader implications of our approach in Section 7.

2. RELATED WORK

The rapid growth of Internet malware, especially in the form of worms and botnets, along with the continuing plague of denial-of-service attacks have led to a great deal of work in both the research and operational community. Early, high-profile distributed denial-of-service attacks, mounted largely with spoofed source addresses, spurred the development of a number of traceback techniques that could enable operators to determine the true source of attack traffic [16, 18, 24].

Unfortunately, spoofed flooding attacks were merely the first salvo in an arms race between motivated cybercriminals and network operators. In an effort to simultaneously obscure their identity and increase the number of potential attack sources available to them, attackers are now recruiting third-party end hosts to construct large-scale botnets, reported to number in the hundreds of thousands. These botnets are then used to launch extremely distributed, large, and pernicious focused denial-of-service attacks [19, 20].

The key aspect of these botnet-powered attacks is the use of third-party computing resources. Hence, the foremost security concern of today's network operators is to defend their networks against attacks launched from an ever-increasing set of well-intentioned but poorly secured networks that fall prey to botnet infiltration. Because these networks are so numerous, various researchers have proposed to fundamentally restrict the Internet's default best-effort any-to-any service model using a variety of technologies, including capability models [7, 15, 25], proof-of-work schemes [12] and, more recently, a "default off" communication model [6]. The attractiveness of the default off model lies in its simplicity: if an Internet host cannot contact you, it can do you no harm (through the network, anyway). Obviously, the effectiveness of all these schemes depends critically on the ability to *a priori* identify hosts that should be allowed to communicate. Unfortunately, prior schemes are either application-specific, require manual configuration, or insert intrusive interactive mechanisms such as CAPTCHAs [12, 22].

Of course, separating the wheat from the chaff is not a new problem. Similar issues arise in the context of developing white and black lists, SPAM filtering, and peer-to-peer reputation systems. The key distinction of our work is the lack of dependence on any particular application or protocol. We explore the potential of generating communities of interest based only upon communication patterns observable inside the network. Community-of-interest-based filtering is complementary to most other mitigation mechanisms, including traditional content-based techniques [17]. We envision that networks may employ PRIMED as a first line of defense, buying much needed time so that targeted, heavyweight approaches can be configured.

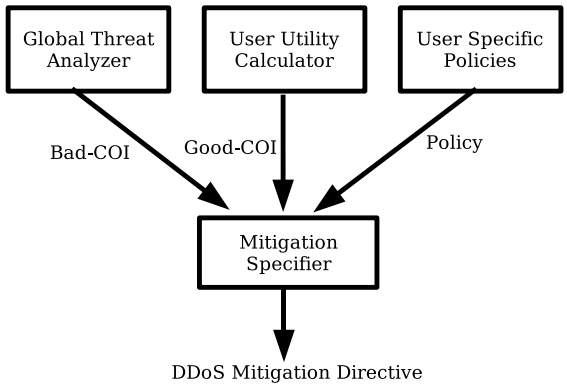

Figure 1: PRIMED architecture

Finally, while we focus on monitoring attack traffic inside a network backbone, a number of alternatives exist. In principle end-users can share information regarding malware, *e.g.*, by sharing past bad behavior using a distributed database [4], or by sharing their firewall logs using an open service such as DShield [2]. In practice, however, these approaches are difficult to realize because of authentication and trust concerns. These concerns are largely avoided in our approach because there already exists a trust relationship between service providers and their customers.

3. ARCHITECTURE

Figure 1 depicts the PRIMED (Proactive, Informed Mitigation of DDoS) architecture. First, an ISP-wide threat analyzer determines the relative security threat posed by certain parts of the Internet and produces the bad COI. This threat analysis is agnostic to attack targets, instead recording only that unwanted traffic from a certain source was detected in the network. Pooling the knowledge regarding unwanted behavior in this way allows customers to benefit from the collective bad behavior observed in the network. Documented evidence of the repeated use of botnets [20] and operational experience with recurring attacks launched from certain countries or ISPs augurs well for the potential benefit of this approach.

Second, a customer-utility calculator determines the parts of the Internet with which each customer of the ISP is likely to have legitimate communication and produces a set of good COIs, one for each customer. The utility of communication by necessity has to be determined on a per-customer basis. For example, for an e-commerce site the utility of repeat "spenders" may be higher than repeat "browsers," and the utility of the latter will in turn be much higher than any attack. Several options exist for constructing good COIs. Because it sees all traffic for its (singly-homed) customers, the provider can again be responsible for determining this set, for example by using flow information. Ideally, however, a customer would construct its own good COI from application-specific information at its disposal and make this available to the provider in some form.

COI membership is combined using customer-specific policies to determine mitigation directives that specify how to treat traffic directed towards a particular customer. How customers want to treat traffic in the different sets will differ depending on their business models and how they use the Internet. For example, an e-commerce site that attempts to attract as many new customers as possible can hardly afford to block traffic from potential new customers. Such a customer might therefore opt to treat traffic from the good COI preferentially, while still admitting all other traffic, even

traffic originating from the bad COI. On the other hand, a business that provides Web-based services to a set of known clients, *e.g.*, outsourced help-desk or human-resource services, can be much more aggressive about limiting access.

From an operational point of view, a straightforward deployment would continually derive good and bad COIs and periodically update the mitigation directives, *e.g.*, on a daily or hourly basis. More sophisticated operational models are definitely possible, for example, when our system is used in combination with reactive mitigation strategies. However, we limit our current focus to simple periodic deployment and the proactive protection that it offers. In the context of this straightforward approach, we seek to achieve the following two operational goals:

Non-interference: Since we advocate a proactive approach to DDoS mitigation it is absolutely essential that there be no interference between customers. In other words, if one customer desires to constrain communication with a particular network entity, that decision should not have any impact whatsoever on the ability of other customers to communicate with said entity. Further, assuming two customers have identical good and bad COIs, they might require completely different policies in terms of how mitigation should be applied.

Correctness tolerance: Our approach effectively constrains any-to-any connectivity of the Internet based on derived COIs. Since even the best of systems makes errors it is critical that our system be tolerant to incorrect classification of network entities. In pursuit of this goal, PRIMED should not impact communication unless access link capacity is constrained.

3.1 Realization options

In this paper we realize our architecture using a classifier, prioritizer, and policer such as depicted in Figure 2, implemented at the ISP's access router. While PRIMED could be implemented based on a coarse-grained pass-or-block policy (*e.g.*, controlled route anouncement), we elect to use more sophisticated QoS mechanisms [10, 23] that allow for differentiated treatment of packets, We thus ensure that our proactive mitagator does not drop traffic unless access link capacity is exceeded, enhancing the correctness tolerance of our approach. Although QoS is not widely deployed today, the QoS *mechanisms* required by PRIMED are commonly available in modern routers.

Deployment of PRIMED along the perimeter of the provider network would minimize the potential for collateral damage within the provider network, but requires distributing customer-specific policies to these routers. While we expect modern routers to be able to deal with the resulting large-scale filtering requirements, we leave a rigorous analysis to future work and focus here on classification and filtering at the provider egress router, thereby protecting customer access links. Preliminary lab testing with edge router platforms indicate that existing routers are able to deal with this more constrained problem without performance degradation.

In this study we construct our COIs based on network prefixes constructed from source IP addresses, hence the possibility of attack traffic using spoofed IP addresses is a concern. Anecdotal evidence [1] suggests that spoofing is not prevalent in today's DDoS attacks, and this claim is further supported by a recent study [13]. Furthermore, a future widespread deployment of ingress filtering [5] would contain spoofing mostly within netblocks, which is adequate for a prefix-based approach. Finally we note that history-based classification does not require the absence of spoofing — it merely requires a degree of predictability in the source IP addresses of attacks. The evaluations we present demonstrate sufficient predictability in the attack traffic we observed.

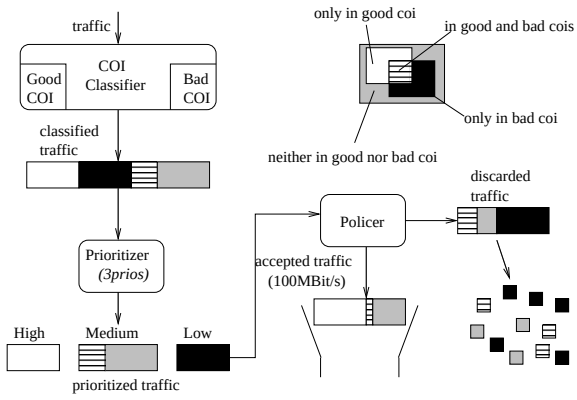

Figure 2: Traffic classifier model

Policy	COI	Prio ←Traffic source
goodcoi	Good	High←good COI
		Low ←remainder
badcoi	Bad	High←remainder
		Low ←bad COI
3prios	Good	High←good COI − bad COI
	Bad	Med ←remainder
		Low ←bad COI − good COI
puregoodcoi	Good	High←good COI − bad COI
	Bad	Low ←remainder
purebadcoi	Good	High←remainder
	Bad	Low ←bad COI − good COI
none	none	Med ←All

Table 1: Simulated customer policies

3.2 Customer policies

Table 1 lists a number of potential prioritization policies that a customer might select based on its network or business needs assuming the three-tiered priority scheme shown in Figure 2. This list is by no means complete, rather we chose a number of simple policies that can be readily implemented to investigate the utility of our approach. The *goodcoi* and *badcoi* policies represent what a customer or ISP might do if they had access to either the good COI or the bad COI, but not both (*e.g.*, if they only wanted to compute one). If we do assume access to both good and bad COIs, we can divide traffic sources into four subsets, as indicated in Figure 2, and can therefore define up to four priority classes. *3prios* considers "good and bad COI" and "good nor bad COI" to be equivalent. Many other mappings of the four subsets to priority classes are possible. We picked two additional policies (*puregoodcoi* and *purebadcoi*) that might be implemented by a mitigator based on passing/blocking rather than prioritization of traffic. For such a mitigator the *puregoodcoi* policy is the most exclusive policy, and would be used by a customer wishing to protect itself from attack traffic at the cost of sacrificing a relatively large amount of good traffic. In the most inclusive policy, *purebadcoi*, the customer wishes to minimize the risk of losing good traffic, at the expense of receiving a large volume of bad traffic. Note that since our model is based on prioritization, *puregoodcoi* and *purebadcoi* represent coarser-grained versions of *3prios* and are expected to perform better than *3prios* only when the good COI and bad COI are highly polluted. Finally, the *none* policy provides a baseline for comparison. We evaluate the effectiveness of these potential customer policies later through simulation.

4. EXPERIMENTAL SETUP

We evaluate the potential of the PRIMED approach in a case study based on traffic traces collected from a tier-1 ISP in 2005. In particular, we obtain NetFlow records related to actual DDoS attacks in the ISP during three months, as well as detailed packet traces of all traffic at a particular data center over a two-week period. We use these traces to derive good and bad COIs and analyze various properties of these COIs in Section 5. In Section 6 we use the same traces to quantify the ability of our approach to protect customers in the data center.

4.1 Data source for wanted traffic

Using the Gigascope monitoring device [9], we collect customer data from a data hosting center in an attempt to construct a pool of desirable traffic for customers in the data center. The data center hosts servers belonging to a variety of companies, ranging from small software development companies to large enterprises such as a content distribution network and a multinational entertainment technology manufacturer.

Before attempting to determine which traffic is wanted, we construct a coherent set of traffic flows for each customer in the data center. If a customer is multi-homed to another provider, we may not be able to see all traffic to or from the customer. Therefore, we select the subset of observed customer prefixes that are announced exclusively through the data center and ignore traffic that is local to the data center. Barring routing changes by the customer or inter-domain routing failures elsewhere, this ensures that we consistently see either all or no traffic from a particular source to the customers. For the other direction we verify that customers mostly respond to traffic using the same provider it arrived from.

Determining a subset of customer traffic that we can reasonably assume is wanted is somewhat trickier. While it is unlikely to accurately capture all of the desired traffic, we use the subset of a given customer's traffic that we can infer as legitimate HTTP request traffic. We make the simplifying assumption that HTTP responses generated by the customer correspond to legitimate requests. An HTTP-based attack requires a TCP handshake, and therefore a greater investment on the part of the attacker than performing a lower-level attack. Therefore we assume that HTTP responses from the customer are responses to legitimate HTTP requests. We realize this is not necessarily true, *e.g.,* it has been observed that attackers attempt to mimic flash crowds [12]. However, we emphasize that our system does not require the COIs to be determined with total accuracy and improving this accuracy (*e.g.,* by using other sources to derive the good COI) will simply improve our system. We can also safely assume that HTTP responses from a customer are not themselves part of an attack that the customer (or a customer of a customer) is launching, as unsolicited HTTP responses are not particularly effective for scans or DOS attacks: since no TCP connection is opened, the receiver ignores them at little cost.

We derive the *good traffic dataset* as follows. For each HTTP response we capture, we locate the flow record of the corresponding HTTP request and add that to the good traffic dataset. For the period we measured, we extracted 240 million HTTP messages per day on average. About 3% of these were excluded using the criteria described above or did not match an HTTP request flow within a 1–2 hour time window.

4.2 Data source for unwanted traffic

In contrast to the traffic used to construct the good COI, we collected unwanted traffic on an ISP-wide basis; we use data associated with DDoS attacks observed in the same tier-1 ISP to derive the unwanted traffic set. A commercial DDoS detection system is deployed at key locations in the ISP and generates alarms based on detected flow anomalies. To ensure comprehensive network coverage, we use the target prefixes identified by these alarms as triggers to collect smart-sampled NetFlow data from the complete ISP network towards the target. NetFlow data is collected for the duration indicated in the alarm plus 30-minute periods before and after so that we can capture any startup/trailing effects.

We next remove flows that would have been discarded by ingress filtering as follows. First, using routing information as well as the NetFlow collection point in the network, we determine the ingress router and ingress interface at which the flow entered our network. If this modeling indicates that the source address in the flow could not have entered the network at the collection point we discard the flow for our analysis. Further, we discard all flows whose source IP address is not routable. Since we collect all NetFlow data towards the target prefixes, these flow records do not necessarily correspond to actual attacks and are likely to have legitimate traffic interspersed. We therefore apply a set of fairly conservative heuristics to identify a subset of flows that are likely sources of attack. The heuristics are as follows. For flows for which at least 3 packets were sampled we estimate a minimum packet rate based on the flow duration and the minimum number of bits that a packet of that protocol type would contain. If this minimum packet rate exceeds the capacity of the egress link towards the target prefix, we consider the flow to be part of an attack. We apply these rate heuristics to all three dominant protocols (TCP, UDP and ICMP). Second, a TCP flow record contains the logical "OR" of the TCP flags of all sampled packets. If the flag combination is illegal or indicates that the TCP flow is likely part of a SYN-flooding attack or reflector attack, then we consider the flow to be part of an attack.

5. COI PROPERTIES

The underlying assumption of our approach is that past behavior predicts future behavior. In this section we explore this assumption for both good and bad behavior by deriving COIs and evaluating their predictability, size and overlap. Predictability quantifies the degree to which a COI generates true positives and false negatives. Specifically, we measure predictability as the fraction of good (bad) traffic senders that are present in the good (bad) COI. Those that are present were correctly predicted by the COI and constitute true positives; those not present in the COI were not predicted and are false negatives. False positives, the counterpart of false negatives, appear in the intersection between a good and a bad COI. Other contributing factors to the intersection are the set of hosts that display both good and bad behavior, and the granularity of the good and bad COIs in terms of prefix lengths used. A COI-based mitigator cannot accurately classify traffic that is in the intersection as either good or bad, and we may expect its performance to be degraded if the intersection is relatively large. Finally, COI size is important for several operational reasons: (1) it determines in part how much space is required to implement a mitigation scheme, (2) it affects the amount of work needed to keep a COI up to date, and (3) it affects the time complexity of online lookup operations. In all cases, a smaller COI results in better performance.

5.1 Properties of good traffic (good COI)

To evaluate the sizes of good COIs, we use the entire period of data center traffic (12 days) to construct a customer's COI. 108 customers have a zero-size COI (*i.e.,* showed no activity in the good traffic data set during our monitoring period) and are omitted from the remainder of our study. There is a wide variety of COI sizes among the remaining 39 customers, ranging from 1 to

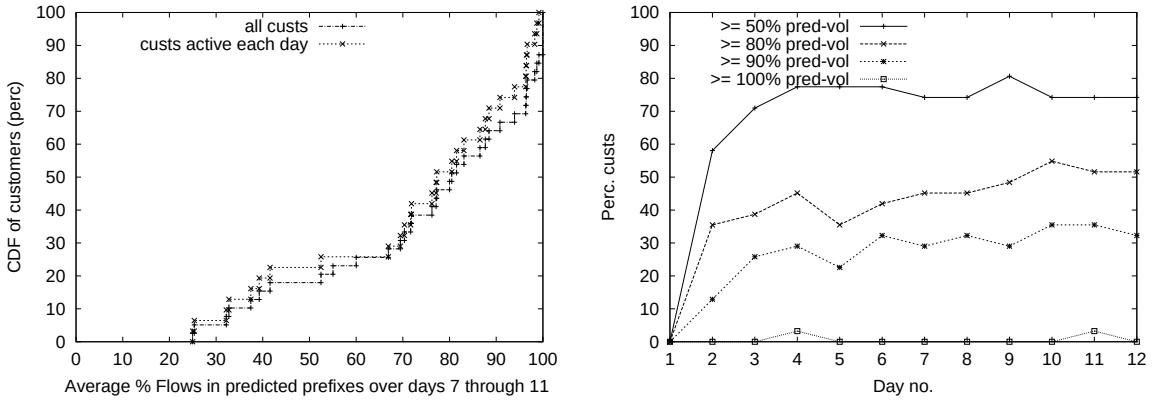

Figure 3: (a) Average daily traffic volume (HTTP transactions) captured by the good COI after the first week. *all custs* plots the 39 customers that had non-zero COI over both weeks, and assumes a value of 100% if a customer had no HTTP traffic on some day. *custs active each day* only plots the 31 customers that had HTTP traffic on all days (including the first week). (b) Daily traffic volume (HTTP transactions) captured by the good COI during both weeks. Only the 31 customers that had HTTP traffic all days are shown.

over 1,000,000 /24 prefixes[1], likely reflecting the relative popularity of Web servers operated by the respective customers. Fortunately, most customers have a small COI: half of the active customers have COI size just over 3,200, and about 75% of the active customers have COI size less than 30,000. These COI sizes are small enough to allow a customer whose access link is under attack to continue to update its COI using a low bandwidth side channel.

Next we examine COI predictability. A customer's good COI *predicted volume* on day d is the fraction of a customer's traffic (in terms of number of good HTTP requests) on day d that is captured by prefixes predicted by the COI computed through day $d-1$. The predicted volume gives a rough indication of the potential performance of a traffic filter based on COIs. Figure 3(a) plots the predicted volume averaged over the four days d in the second week of our trace. We see that all customers have an average predicted volume of at least 25% and the majority of customers have an average predicted volume of at least 75%.

Figure 3(b) shows predicted volume in greater detail. For each day d, it plots the number of customers for which predicted volume was at least p%, where different lines show different values of p. As one would hope, the number of customers with a certain minimum predicted volume increases with time. After the first week, the number of customers with predicted volume $\geq$ 80% is always at least 14 (45%), or, if we include the customers without traffic, 20 (51% of such customers). Of these, the same 12 and 17 customers consistently satisfy the 80% criterion. The lines appear to stabilize, suggesting that there is little point in including more than a week's worth of prefixes in a COI.

5.2 Properties of attack traffic (bad COI)

For the bad COI evaluation, we use data from a 19-week period to determine how well the IP addresses observed on a particular day are predicted by the prefixes that were involved in attacks in previous days. We calculate the effect of the amount of history used by determining for each day N in our data the union of prefixes in the M previous days (up to day $N-1$), for M ranging from 1 day to 100 days. Figure 4 shows the average percentage of IP addresses that matched as a function of the number of days (M) used to determine the bad COI prefix set. Results are shown for traffic before and after our attack heuristics of Section 4.2 are applied. Both

[1] Due to space constraints, we only present results when considering network entities as /24 subnets; results are similar for other granularites [21].

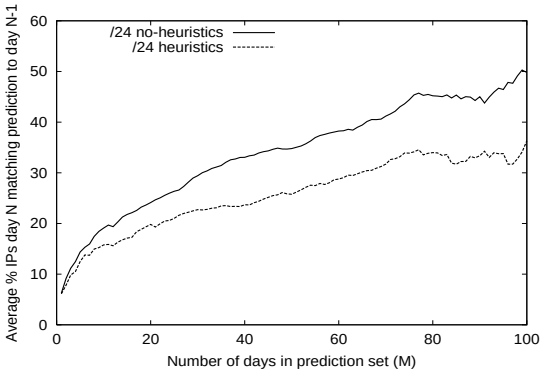

Figure 4: Bad COI predictability: Average percentage of IPs in day N matching a bad COI calculated over the previous M days.

curves show an initial steep increase as we increase the number of historical days, but level off towards the end of the curve. The corresponding average sizes of the bad COI prefix set (not shown) rise to around 10,000 /24 prefixes (heuristics) and 90,000 /24 prefixes (no heuristics). We use a three-month period to construct the bad COI for the evaluation in the remainder of this paper.

5.3 COI intersection

In Figure 5 we plot for each customer the percentage of the good COI that intersects with the bad COI. If we assume that good and bad COIs predict good and bad sources accurately, and that the traffic volume from the good and bad COI is proportional to their sizes, we can distinguish two (overlapping) types of customers. For the customers in the right portion of Figure 5, a small percentage of the good COI intersects. For these customers, using a filter based only on the bad COI would deprioritize all traffic from bad sources, while sacrificing (deprioritizing) traffic from 1–15% of good sources (15% is without the use of attack heuristics). For the twenty customers with smallest sized good COIs a very small percentage of the bad COI intersects with the good COI: no more than 2–9% (not shown). Therefore for these customers a filter based on the good COI alone is likely to perform well, prioritizing all traffic from good sources and deprioritizing traffic from virtually all bad sources.

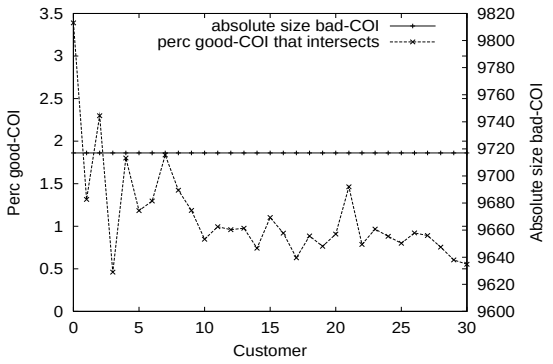

Figure 5: Percentage of good COI that intersects with the bad COI. Customers are ranked in order of increasing good COI size. Only customers with non-empty intersection are shown.

In summary our evaluation of the COI characteristics show that both good and bad COIs show significant predictability. Further, while overlap did occur between the good and bad COIs, this overlap was small for the majority of customers, indicating the potential of our approach.

6. EVALUATION

Having established both good and bad COIs, we are now prepared to evaluate the effectiveness of the PRIMED architecture assuming the three-tiered classifier/policer model as depicted in Figure 2. Operational constraints prevent us from actually deploying PRIMED at the data center, so we present a trace-driven simulation instead. For simplicity we model each customer as having a 100-Mbit/s switched access link, ignore TCP/IP overhead, and assume that the incoming link is only used by good HTTP traffic mixed with attack traffic (*i.e.,* we do not consider the impact of other non-attack traffic). These assumptions serve mainly to focus our discussion; performance results turn out to be insensitive to available link capacity. Conceptually, traffic passes through the traffic filter as follows. First, the COI classifier examines the source address of all traffic arriving at the access link and classifies it into one of the following classes: "only good COI," "only bad COI," "good and bad COI," or "neither good nor bad COI." The prioritizer then prioritizes the traffic classes according to the policy under test. The example in Figure 2 shows the *3prios* policy, which places traffic into three priority classes (see Table 1). Note that *3prios* is just one possible policy; the number of priority classes as well as the mapping from COI classes to priority classes varies from policy to policy. Finally, the policer admits only as much traffic as will fit through the access link while respecting the priorities established by the prioritizer; excess traffic is discarded. For simplicity, we discard equal percentages of good and bad traffic within a particular priority class.

6.1 Simulation setup

Our evaluation consists of directing recorded attacks at each customer in the data center. Since the attacks in our traces are generally not directed at the customers in the data center, we modify the attack traffic so that each attack is directed at each customer in turn.

To conduct a realistic evaluation, we divide our dataset into a training set and a test set. We compute good and bad COIs over the training set and use the good and bad traffic from the test set to drive our simulations. The bad COI training set consists of attacks ranging from 26 May to 24 Aug, 2005. The training set for the good COI consists of the HTTP traffic as described in Section 4 from 18 to 24 Aug. The test traffic is drawn from both the HTTP and attack traffic from 25 to 29 Aug (except 27 Aug which we excluded after discovering a measurement error). Thus we have two bad COIs constructed with and without attack heuristics, and two similarly constructed sets of attack test traffic.

To cut down on computation time, we (a) only simulate what happens while an attack takes place, (b) use per-attack average traffic volumes for each of the subsets in Figure 2, and (c) estimate the average based on further sampling during the attack. Normally we sample the attack traffic at one-minute intervals. However, some attacks are very short and produce few samples (less than five) when sampled at one-minute intervals; these short attacks are sampled at one-second intervals instead. What happens outside of attack periods is not relevant to our study since none of the customers' access links were otherwise overloaded during our observation period.

6.2 Performance results

We ask the question "how many bytes of attack traffic can we send to a customer before the customer loses 5% or more of its HTTP traffic (in bytes)?" Note that in asking this question we are ignoring the actual absolute volume of the attacks in our dataset—we simply scale them up or down accordingly. We do, however, consider the relative volumes of attack traffic in each of the subsets shown in Figure 2. In [21] we derive for each prioritization policy P, the value B_P^*: the minimum amount of bad traffic that causes the customer to lose at least 5% of good traffic.

Figure 6(a) plots B_{none}^*/B_P^*, *i.e.,* the *inverse* of performance improvement of P with respect to having no DDoS attack mitigation. As an example, consider the solid circle at approximately $(0.2, 60)$ in Figure 6(a). This point indicates that in 60% of attacks *3prios* achieved about $(1/0.2 = 5)$-times better performance than doing no mitigation. A value of 1 means performance did not change; a value > 1 means B_P^* did worse than B_{none}^*. A value of 0 (improvement ∞) means that no amount of bad traffic could displace 5% of good traffic, in which case we say there is complete protection. Overall the *3prios* policy performs best: in 38–53% of attacks *3prios* was able to achieve complete protection (38% is without the use of attack heuristics, not shown) whereas in only 7–9% of attacks *3prios* gave the same or worse performance than having no mitigation (*i.e.,* inverse performance improvement ≥ 1). In Figure 6(b) we attempt to characterize the effect of the attacks per customer. For each customer we rank its attacks by inverse performance improvement, and then take the inverse performance improvement of the median attack. Figure 6(a) is a CDF of these median values, and shows that *3prios* gives complete protection on the median attack for 29–44% of customers. From the 5^{th} percentile values (*i.e.,* the attacks for which *3prios* performed best, not shown), we find that *3prios* is able to completely protect every customer during a minimum of 5% of attacks on the customer. For some customers there are attacks that are slightly worse under our mitigation policies than under the *none* policy. They are not apparent in the median plot, but when considering the 95^{th} percentile attacks (*i.e.,* the attacks during which *3prios* does worst, again not shown) this turns out to be an issue for 22–50% of customers using *3prios*. However the worst inverse performance improvement of *3prios* that we observe is no more than 1.06 (5.7% degradation). Table 2 summarizes these figures for *3prios* and the second-best performer, *goodcoi*.

7. CONCLUSION

Our approach to deriving good and bad COIs has been rather straightforward, relying on the use of simple heuristics and /24 pre-

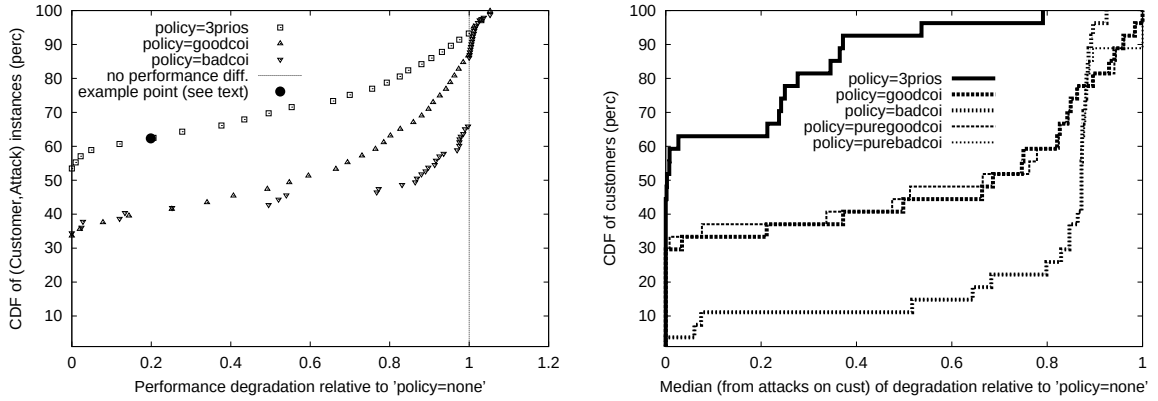

Figure 6: Inverse of performance improvement of DDoS mitigation. (a) All attacks on all customers. (b) For each customer, the median from all attacks on the customer. For clarity not all data points have been plotted. In (a) we ignored (customer,attack) instances where the customer did not receive HTTP traffic during the attack. We also omitted in (a) *puregoodcoi* and *purebadcoi* which are almost identical to *goodcoi* and *badcoi*, respectively. In (b) we ignore customers that did not receive HTTP traffic during at least 20% of attacks (leaving 27 customers), and for each customer only use (customer,attack) instances where the customer received HTTP traffic during that attack.

	3prios	*goodcoi*
attacks with ∞ performance	38-53%	33-34%
attacks with degradation	5-7%	6%
attacks with same performance	2%	7%
customers with ∞ on median attack	29-44%	29%
customers with ∞ on 5^{th} perc attack	100%	93%
custs w/ degrad on median attack	0%	0%
custs w/ degrad on 95^{th} perc attack	22-50%	14-19%
worst-case degradation	5.7%	5.7%

Table 2: Performance improvement over policy *none*.

fixes. Nevertheless, for the dataset discussed we were able to construct COIs of reasonable quality, and our *3prios* mitigation policy combined the strengths of the good and the bad COIs. Of course, our work must be seen as part of an ongoing arms race. While we have demonstrated that PRIMED is able to raise the bar against successful DDoS attack, attackers will inevitibly try to game our system, *e.g.*, by attempting to pollute good and bad COIs and by circumventing simple heuristics using flash-crowd style attacks. However we believe our system is flexible enough to incorporate more sophisticated heuristics for separating good from bad traffic as they become available. Additionally, since we construct COIs off-line, the complexity of such heuristics is of smaller importance to our proactive approach than it is to a reactive mitigation system. As a specific example, a weakness of our current heuristics is that an attacker may launch a spoofed attack in order to incriminate a member of the good COI (the victim) by making it appear as though the attack is originated by the victim. As a result, the victim now also becomes part of the bad COI. A possible improvement would be to "weigh" the evidence for and against an entity before deciding whether it should be part of the good and/or the bad COI.

Another limitation of our approach is that it assumes attacks are sourced by predictable source IP addresses. In particular PRIMED may be ineffective against reflector attacks that abuse different reflectors at different times. Finally, our model assumes that an attack targets an access link or, more generally, provider resources allocated to a *single* customer. In future work we intend to investigate protection of resources shared by multiple customers, as well as efficient implementation of mitigation strategies on the ingress routers into a provider network. Another area for future work is to determine the the appropriate frequency at which good and bad COIs should be updated.

In parting, we observe that our approach raises an important side question, which is whether and how network sources being identified as "considered bad" are notified about the fact that they have been classified as such. There are clearly more than technical issues with this question including business and legal concerns. However, we can make a number of observations. First, in order to allow for corrective action, it seems reasonable to assume that either the list of suspected bad network entities should be made public, or that at least the "owner" of the network entity be notified of the classification. Making this information public (even in a limited way) immediately raises the concern that attackers might discover what network entities are less effective for their purposes, or, more importantly, will know which entities are *not* in that set and would therefore be more desirable for their needs. We argue that as long as mitigation does not involve outright prevention of communication, this approach provides a desirable incentive model for "cleaning up" the Internet. Specifically, the owner of a network entity listed in a bad COI knows that that traffic that it generates might be restricted by some destinations. The owner in question can deal with the cause of the problem and be removed from the bad list. Alternatively, the owner can decide to not deal with the problem, but risk that some of its communication may be impacted.

8. REFERENCES

[1] Communications Innovation Institute. Summary of the initial meeting of the DoS-resistant Internet working group. `http://www.thecii.org/dos-resistant/meeting-1/summary.html`.

[2] Distributed intrusion detection system. `www.dshield.org`.

[3] W. Aiello, C. Kalmanek, P. McDaniel, S. Sen, O. Spatscheck, and J. Van der Merwe. Analysis of Communities of Interest in Data Networks. In *Proc. PAM Workshop*, Mar. 2005.

[4] M. Allman, E. Blanton, and V. Paxson. An architecture for developing behavioral history. SRUTI Workshop, July 2005.

[5] F. Baker and P. Savola. Ingress Filtering for Multihomed Networks. *Internet Engineering Task Force*, March 2004. RFC 3704.

[6] H. Ballani, Y. Chawathe, S. Ratnasamy, T. Roscoe, and S. Shenker. Off by default! In *Proc. ACM HotNets Workshop*, Nov. 2005.

[7] M. Casado, T. Garfinkel, A. Akella, D. Boneh, N. McKeown, and S. Shenker. SANE: A protection architecture for enterprise networks. In *Proc. ACM/USENIX NSDI*, May 2006.

[8] C. Cortes, D. Pregibon, and C. T. Volinsky. Communities of interest. *Intelligent Data Analysis*, 6(3):211–219, 2002.

[9] C. Cranor, T. Johnson, O. Spatscheck, and V. Shkapenyuk. Gigascope: A stream database for network applications. In *Proc. ACM SIGMOD*, June 2003.

[10] M. A. El-Gendy, A. Bose, and K. G. Shin. Evolution of the Internet QoS and support for soft real-time applications. *Proceedings of the IEEE*, 91(7):1086–1104, July 2003.

[11] J. Ioannidis and S. M. Bellovin. Implementing pushback: Router-based defense against DDoS attacks. In *Proc. NDSS*, Feb. 2002.

[12] S. Kandula, D. Katabi, M. Jacob, and A. Berger. Botz-4-Sale: Surviving organized DDoS attacks that mimic flash crowds. In *Proc. ACM/USENIX NSDI*, May 2005.

[13] Z. M. Mao, V. Sekar, O. Spatscheck, J. Van der Merwe, and R. Vasudevan. Analyzing Large DDoS Attacks Using Multiple Data Sources. In *Proc. SIGCOMM Workshop on Large Scale Attack Defense (LSAD)*, Sept. 2006.

[14] P. McDaniel, S. Sen, O. Spatscheck, J. Van der Merwe, W. Aiello, and C. Kalmanek. Enterprise security: A community of interest based approach. In *Proc. NDSS*, Feb. 2006.

[15] B. Raghavan and A. C. Snoeren. A system for authenticated policy-compliant routing. In *Proc. ACM SIGCOMM*, pages 167–178, Aug. 2004.

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

[17] S. Singh, C. Estan, G. Varghese, and S. Savage. Automated worm fingerprinting. In *Proc. USENIX OSDI*, Dec. 2004.

[18] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, B. Schwartz, S. T. Kent, and W. T. Strayer. Single-packet IP traceback. *IEEE/ACM Transactions on Networking*, 10(6):721–734, Dec. 2002.

[19] S. Staniford, V. Paxson, and N. Weaver. How to 0wn the Internet in your spare time. In *Proc. USENIX Security*, Aug. 2002.

[20] The Honeynet Project. Know your enemy: Tracking botnets. `http://www.honeynet.org/papers/bots`.

[21] P. Verkaik, O. Spatscheck, J. Van der Merwe, and A. C. Snoeren. PRIMED: A community-of-interest-based DDoS mitigation system. Technical report, AT&T Labs-Research, Apr. 2006.

[22] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In *Proc. EUROCRYPT*, 2003.

[23] X. Xipeng and L. M. Ni. Internet QoS: A big picture. *IEEE Network*, 13(2):8–18, Mar./Apr. 1999.

[24] A. Yaar, A. Perrig, and D. Song. An endhost capability mechanism to mitigate DDoS flooding attacks. In *Proc. IEEE Security and Privacy*, May 2004.

[25] X. Yang, D. Wetherall, and T. Anderson. A DoS-limiting network architecture. In *Proc. ACM SIGCOMM*, Aug 2005.