LADS: Large-scale Automated DDoS detection System

Vyas Sekar
Carnegie Mellon University

Nick Duffield
AT&T Labs-Research

Oliver Spatscheck
AT&T Labs-Research

Jacobus van der Merwe
AT&T Labs-Research

Hui Zhang
Carnegie Mellon University

Abstract

The last few years have seen a steady rise in the occurrence and sophistication of distributed denial of service (DDoS) attacks. Volume-based attacks aggregate at a target's access router, suggesting that (i) detection and mitigation is best done by providers in their networks; and (ii) attacks are most readily detectable at access routers, where their impact is strongest. In-network detection presents a tension between scalability and accuracy. Specifically, accuracy of detection dictates fine grained traffic monitoring, while performing such monitoring for the tens or hundreds of thousands of access interfaces in a large provider network presents serious scalability issues. In this work we investigate the design space for in-network DDoS detection and present a triggered, multi-stage approach that addresses both scalability and accuracy. Each successive stage can access finer resolution data sets, and can perform deeper, more expensive diagnostics if required. We argue that this approach is applicable to any economically feasible, large scale, DDoS detection system. Our second contribution is the design and implementation of an operational instance of our triggered, multi-stage approach. The attractiveness of this system lies in the fact that it makes use of data that is readily available to an ISP. Specifically, SNMP-based anomalies trigger the collection of Netflow data for detailed attack analysis. Aggregation and compression on the flow data is used to generate alarms concerning possible attack targets. We evaluate the system using SNMP and Netflow data collected from a large tier-1 ISP and compare the results with alarms generated by a commercial DDoS detection system. Our triggered approach achieves high accuracy with fairly modest processing requirements.

1 Introduction

Targeted Denial of Service and Distributed Denial of Service attacks are continuously on the rise in the last few years. Armies of botnets comprised of compromised hosts can be utilized to launch attacks against specific Internet users such as enterprises, campuses, web servers, and homes. In this paper, we focus on an important class of DDoS attacks, namely, brute force flooding attacks. We observe that access links are typically the bottleneck link for most Internet users, and that an attack can easily send sufficient traffic to a user to exhaust its access link bandwidth capacity or overload the packet handling capacity of the routers on either end of the link [7].

Brute force flooding attacks are easy for attackers to launch but are difficult for targeted users to defend, and therefore represent a clear and present threat to Internet users and services. Given the limited capacity of most access links on the Internet, a successful DDoS attack may only involve relatively few attack sources. In addition, the size of some reported attack networks [25] suggests that a determined attacker might be capable of overloading even the largest access links. From a user's perspective, a bandwidth attack means its *in-bound* capacity is exhausted by the incoming attack. Given that a user often controls only one end of the access link, for example via a Customer Equipment or CE router (see Fig. 1, while its ISP controls the other end (referred to as C-PE, or Customer-Provider Edge router in Figure 1), once an access link is overloaded there is precious little that the target of the attack can do without the assistance of its ISP. In fact, even automated DDoS related mechanisms originating from the customer side of an access link becomes useless once the access link itself is overloaded.

For these brute force bandwidth attacks, we therefore reason that a very promising architecture is one that performs *in-network detection and mitigation* of DDoS attacks by the service provider. With respect to mitigation, since it is *upstream* of users' access links, an ISP may defend against bandwidth attacks by deploying appropriate network filters at network routers to drop malicious packets, or alternatively using routing mechanisms to filter packets through scrubbers [1]. With respect to de-

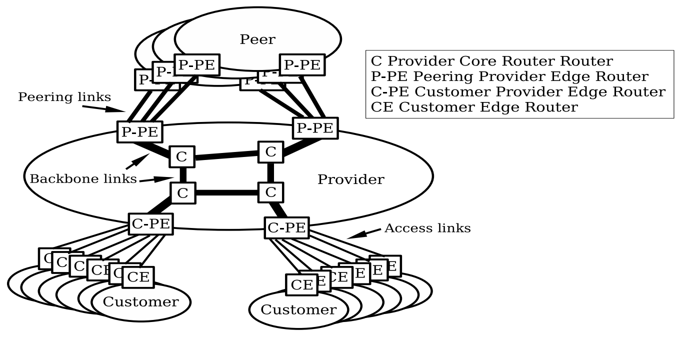
Peer
P-PE
C Provider Core Router Router
P-PE Peering Provider Edge Router
C-PE Customer Provider Edge Router
CE Customer Edge Router
Peering links
Backbone links
C
Provider
C-PE
Access links
CE
Customer

Figure 1: Components of a Provider Network

tection, the key challenge is to come up solutions that satisfy multiple competing goals of scalability, accuracy, and cost-effectiveness. Specifically, accurate detection demands fine grained traffic monitoring, while performing such detection and monitoring for the tens or hundreds of thousands of access interfaces in a large provider network presents serious scalability and cost issues.

In this paper, we present a general framework for a triggered multi-stage infrastructure for detection and diagnosis of brute-force flooding Denial of Service attacks. The first level of the multi-stage approach consists of a low cost anomaly detection mechanism that can provide information to traffic collectors and analyzers to reduce the search space for traffic analysis in both temporal and spatial dimensions. Successive levels of the triggered framework, invoked on-demand and therefore *much less frequently*, then operate on data streams of progressively increasing granularity (e.g., flow level or packet level traces), and perform detection and diagnostic methods of increasing computational cost and complexity. This architecture fits well with the hierarchical and distributed nature of the network. The first stage requires simple enough processing capability to be implemented in distributed fashion for all customer-facing interfaces. The later processing capabilities can be more centralized and thus shared by many edge routers.

We have designed and implemented an operational DDoS detection system based on the triggered multi-stage architecture within a tier-1 ISP. Our system makes use of two sources of data: SNMP and Netflow, both of which are readily available in commercial routers today. We adopt a two-stage approach in our system. In the first stage, we detect volume anomalies using simple low-cost SNMP data feeds such as packets per second. These anomaly incident reports are then used to trigger flow-collectors that collect Netflow records for the appropriate routers, interfaces, and time periods. We then perform automated analysis of the flow information using uni-dimensional aggregation and clustering techniques. There are advantages of using only data sources that are readily available. For example, given the ubiquitous availability of SNMP data for all router interfaces, our system is able to perform the first stage analysis directly on traffic rates of all customer-facing interfaces, which, by their bottleneck nature, are those most affected by attacks. As a comparison, many commercial DDoS offerings, including the one discussed later in this paper, deploy monitoring solutions in the core or at the peering edges in order to minimize the number of monitoring devices (thus cost) required. Such schemes are likely to miss smaller attacks which, while large relative to the targeted interface, are small amongst aggregate traffic in the core. In contrast, our system has ubiquitous monitoring but no additional cost, and can perform anomaly detection considering both traffic volume and link speed for all customer-facing interfaces.

There are two key contributions of our work. First, we propose, design and evaluate a triggered infrastructure that meets the constraints of operational cost and complexity, and also substantially raises the bar in terms of detection and diagnostic capabilities with respect to DoS attacks. Putting the pieces together to build such an operational system is a non-trivial task. Second, we believe that the system that we have designed can be used as a well-documented, vendor-independent benchmark, to quantify the extent and magnitude of attacks, using data that is readily available to most providers.

We review the related work in Section 2 before describing our architecture for scalable DDoS detection in Section 3. Section 4 describes the specific implementation based on collection of Netflow records triggered by SNMP anomalies. We evaluate the performance of the system in Section 6, demonstrating both the efficacy and the operational feasibility. We conclude in Section 7.

2 Related Work

In the context of triggered measurements ATMEN [14] provides a general communication framework. Our work could utilize such a framework if available, however, ATMEN does not address the detection of DDoS attacks.

The spectrum of anomaly detection techniques ranges from time-series forecasting [5, 24] and signal processing [4], to network wide approaches for detecting and diagnosing network anomalies [15, 32]. These approaches are intended for detecting coarse-grained anomalies, and do not necessarily provide the diagnostic capability required for large-scale DDoS detection.

Also related to our multi-stage approach are techniques for fine grained traffic analysis, using flow or packet-header data. Techniques for performing detailed multi-dimensional clustering at scale [9, 30] to generate concise traffic summary reports, are of particular interest to attack detection. Other solutions for online traffic analysis use either optimized data structures and/or counting algorithms [33, 13, 10] for detecting heavy-hitters.

Moore et al [21], detect attacks utilizing the fact that many types of attacks generate *backscatter* traffic unintentionally. Network telescopes and honeypots [29] have also been used to track botnet and scan activity. Some of the early DoS attacks typically used source address spoofing to hide the sources of the attacks, and this motivated a large body of literature on IP traceback. Proposed solutions include packet marking [26, 23], hash-based traceback [27], and reverse path flooding [6].

There are also several proprietary DDoS detection systems available [3, 19]. In our evaluation, we compare the set of alarms generated by LADS with those generated by a commercial system deployed at a Tier-1 ISP.

There are several recent proposals for mitigating DoS attacks. Many solutions rely on infrastructural support for either upstream filtering [17], or using overlays [12, 28]. Recent work also focuses on re-designing networks providing a framework of capabilities to prevent flooding attacks [31, 2]. Several proposals [11, 22] focus on end-system solutions combining specific types of Turing tests and admission control to enable servers to deal with flooding attacks and flash crowds gracefully. Mirkovic et al. [20] provide an excellent taxonomy of DDoS attacks and defenses.

3 Scalable In-Network DDoS Detection

Having argued for the necessity of in-network DDoS detection (and mitigation) in Section 1, we now consider the implications of this approach for building a detection system in a large provider network. Like any anomaly detection system the main requirement for LADS is accuracy, i.e., having a low false alarm and miss rate. The second requirement is timeliness: to be of practical value a detection system should provide near real time detection of attacks to allow mitigation mechanisms to be applied. Third, to be useful a detection system should cover all (or most) customers of a provider. The number of customers could range from hundreds for a very small provider to hundreds of thousands for large providers.

These requirements have significant system scalability implications: (i) Is it feasible to collect detailed enough information to allow attack detection on a per-customer basis? (ii) Is it feasible to perform in timely fashion the processing involved with the detection on a per-customer basis? Next we present our triggered multistage DDoS detection approach and explore several possible implementation alternatives.

3.1 Triggered Multistage DDoS Detection

There are two sources of complexity and problems of scale in our context: *Collection* and *Computation*. The collection complexity arises from the fact that data streams have to be selected from monitoring points (links/routers), and either transported to an analysis engine (possibly centralized) or provided as input to local detection modules. The computation complexity arises from the algorithms for analyzing the collected data, and the sheer size of the datasets. We observe that not all detection algorithms have the same complexity: the differences arise both from the type of data streams they operate on and the type of analysis they attempt to perform.

As a simple example, consider two types of data streams that are available from most router implementations: simple traffic volume statistics that are typically transported using SNMP, and Netflow like flow-records. From an operational perspective, enabling the collection of the different data sets on routers incurs significantly different costs. There are three main cost factors: (i) the memory/buffer requirements on routers, (ii) the increase in router load due to the monitoring modules, and (iii) bandwidth consumption caused by transporting the measured data. The SNMP data has coarse-granularity and the typical analysis methods that operate on these are lightweight time-series analysis methods [5, 24, 4]. The Netflow data contains very fine grained information, and as a result is a much larger dataset (in absolute data volume). It is easy to see that the flow data does permit the same kind of volume based analysis that can be done with the SNMP data set by simply aggregating the flow data into traffic counts. However more powerful analysis can extract greater benefit of the finer-granularity data using sophisticated algorithms [9, 30], for culling out traffic patterns that are “interesting”.

The presence of heterogeneous data sources which offer varying degrees of detection power at different computation and collection costs raises interesting design questions. At one extreme we could envision running sophisticated anomaly detection algorithms on the fine granularity data (i.e., Netflow) on a continuous basis. The other extreme in the design space would be an entirely light-weight mechanism that operates only on the coarse-granularity data. Both these extremes have their potential pitfalls. The light-weight mechanism incurs very little computational cost, but potentially has to make compromises on the enhanced diagnostic capabilities that the fine grained analysis provides. A key observation in this regard is that close to the attack target, e.g., at the customer access link, detection of brute force flooding attacks become reasonably easy, although it might generate false alarms, and lack the ability to generate diagnostic attack reports of use to operators. The heavy-weight mechanism, on the other hand, incurs a much higher collection and computation cost. Also such mechanisms potentially operate in an *unfocused* manner, i.e., without knowledge about the seriousness of the incidents that actually need

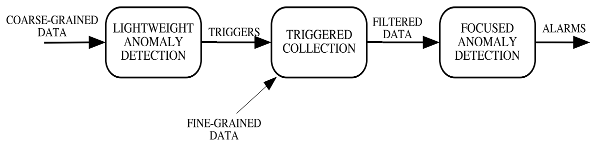

Figure 2: Triggered Multistage DDoS Detection

operators' attention. Operating in such an agnostic setting may in fact be as detrimental to the accuracy as the false positive rate may be intolerably high.

Our work attempts to find a operationally convenient space between these extremes. The key idea in our approach is to use *possible* anomalous events detected in coarse grained data *close* to the attack target, to *focus* the search for anomalies in more detailed data. Specifically we propose a triggered multistage detection mechanism in which the successive stages can have access to and operate on data streams of increasing granularity. Potential anomalous events in earlier stages serve as triggers for later stages to selectively collect more fine grained data on which more sophisticated anomaly detection can be performed. This approach is depicted in Figure 2. Intuitively the triggered approach helps focus our resources intelligently, by performing inexpensive operations early on, while allowing for sophisticated, data and compute intensive tasks in the later stages of the multi-stage triggered approach to get better incident diagnostics.

3.2 Design Alternatives

While our approach generalizes to any number of detection stages, the system described in this paper is limited to two stages. Specifically, for the lightweight anomaly detection we make use of *volume anomaly detection* on SNMP data. For the second stage we use a combination of *rule-based detection and automated uni-dimensional clustering* on sampled Netflow data. In this section we briefly describe our selected methods as well as other potential alternatives.

3.2.1 Lightweight Anomaly Detection

As the name suggests the key constraint here is that the method not require significant processing so that it can be applied to a large number of interfaces. Since the output of this stage triggers more detailed analysis in the second stage, false positives are less of a concern than false negatives. In other words, a false positive from the first stage will only cause more unnecessary work to be done in the second stage (and hence not necessarily much of a concern for the operator who only sees the final alarms after the second stage), whereas a false negative would cause an attack to be missed altogether.

Volume anomaly detection: Traffic anomalies on volume and link utilization data available from egress interfaces, are often good indicators of flooding attacks, and hence can be used to trigger further investigation. Metrics of interest, that are available from most current router implementations, include the traffic volume (either in bytes per second or packets per second), router CPU utilization, and packet drop counts. Our implementation uses volume anomalies on the packets per second and is described in detail in Section 4.1. The basic idea behind our approach involves the use of a time series of link utilization to model the expected future load on the link and to then detect significant deviations from this expected behavior.

Using traffic distribution anomalies: Lakhina et al. [16] discuss the use of distributions, specifically using the entropy of distributions for diagnosing anomalies in networks. The key idea here is that many common attacks can be identified by substantial changes in specific distributions, such as the traffic observed on source and destination addresses and ports. While the use of distribution information in their work was primarily to augment volume anomaly detection, we can use the distributions as a trigger for further analysis. The use of such more informative metrics for diagnosis, may not necessarily reduce the collection cost, since computing these metrics may in fact need access to very fine-grained traffic information, but they can potentially reduce the computation cost.

3.2.2 Focused Anomaly Detection

Even though our staged approach will reduce the search space for the second stage detection significantly, the scale of the problem is such that computational overhead remains a concern. Of course accuracy of detection is the other main criteria for this stage.

Rule-based detection: It is well-documented that most common denial of service attacks fall under a small number of categories, that can be easily captured with a small set of rules for detection. For example, a large number of ICMP ECHO or TCP SYN messages going to a single destination IP address is often indicative of a flooding attack. Another option is to use *botnet blacklists* to check if the set of source addresses that occur frequently in the traffic belong to known compromised machines (referred to as zombies) used for launching attacks. Rule-based approaches have near-term appeal since they typically have low false-positive rates, even though their detection capabilities are limited to the set of attacks spanned by the rule-sets.

Automated Uni-dimensional clustering: Our specific implementation for the second stage involves the use of uni-dimensional functionality of hierarchical aggregation algorithms. Conceptually, uni-dimensional clustering attempts to discover heavy-hitters along source/destination prefixes, using some thresholding scheme to compress reports along the prefix hierarchy. Since the computational overhead with performing uni-dimensional aggregation is fairly low, and the functionality provided is sufficient for investigating most known types of DDoS attacks we choose this approach. Our implementation, which can be viewed as a combination of uni-dimensional clustering and rule-based approaches, is described in detail in Section 4.2.

Automated Multi-dimensional clustering: Multi-dimensional clustering provide another alternative for the second stage detection [9, 30]. The basic theme of these approaches is to abstract the standard IP 5-tuple (srcaddr, dstaddr, protocol, srcport, dstport) within multi-dimensional clustering techniques to report traffic patterns of interest. Typically, the size and complexity of the generated clusters can be reduced substantially by tuning the technique to report only interesting clusters, those that either have a high volume or those that have a significant deviation from an expected norm [1].

3.3 Benefits and Pitfalls

In this section we discuss some of the benefits and potential pitfalls of our approach. We first presents the benefits of our triggered approach.

Detecting high-impact attacks – Since our triggers are generated close to the customer egresses, we are more likely to detect attacks that actually impact the end-user. Note that this is in contrast to more centralized approaches, even those that work on more fine-grained data feeds [2]. For example, by monitoring SNMP byte counts on a T1 access link it is straight forward to determine when the link is being overloaded. Looking for the same information from a different vantage point, e.g., at a set of major peering links is a much more challenging task. Not only could the traffic flowing towards the T1 interface be spread across many such peering interfaces, but the traffic will be hidden inside an overwhelming amount of other traffic on the peering links.
Efficient data collection – The fact that our approach works with different data sources of differing granularity also has the advantage of efficient data collection. Specifically, for our implementation where we use SNMP data for the first stage detection, this data is lightweight enough that it can be collected on the access routers without imposing significant load. The Netflow data, on the other hand, can be more efficiently collected at more powerful core routers so that access routers are not burdened with this more heavy weight process, especially when the access router is under attack.
Reduced computation cost – We use high cost operations and expensive algorithms in a focused manner, and also significantly reduce the data volumes that the expensive operations need to handle.
Low operational complexity – The different stages are fairly simple and easy to understand, and vendor-independent. Managing the operation should be really simple. More importantly, our implementation works with data streams that are already available to most provider networks, and deploying our design does not incur any overhead in terms of instrumenting new monitoring capabilities or deploying special hardware for collection and analysis.
Near real-time incident reports – Since the computational complexity is significantly reduced, we can operate the system in near real-time, without dependence on specialized hardware or data structure support.
Flexibility – Our approach is flexible in two aspects. First we can easily accommodate other data streams as and when they are available. Second, within each stage the performance and algorithms can be optimized to reach desired levels. For example, our first stage triggers currently use simple time-series volume anomaly detection. It is fairly easy to augment this step with other data streams, and traffic statistics, or alternatively use other anomaly detection methods for the same data stream.

In our triggered approach, there are three potential pitfalls. The first pitfall is one relating to possible undesirable interactions between the trigger stage and the detailed analysis stage. While our approach allows for each component to be optimized in isolation, optimizing the overall system performance would require a detailed understanding of the interfaces and interactions between different components. Managing and optimizing such multi-component systems is inherently complicated – we believe our specific implementation is based on a clean set of interfaces, are sufficiently decoupled, and hence have very few undesirable interactions.

The second, more serious problem, is one of misses due to the triggered approaches. While the triggers reduce the operational complexity, they may be doing so by compromising the sensitivity of the system, i.e., by increasing the false negative rate. Attacks that can cause the greatest disruption in terms of traffic engineering, routing etc., are volume attacks, which will invariably show up as volume anomalies on the egress interfaces closest to the customers. Since our focus is on such flooding attacks,

[1]Despite this tunability, our experimentation with the publicly available multi-dimensional clustering tool, Autofocus [9], showed that this approach was still too compute intensive for the volume of data to be analyzed.

[2]Due to cost and operational constraints commercial vendor detection systems are typically constrained to operate in such a centralized model, using feeds near the core of the network.

there is almost no impact on the false negative rate. By only looking at volume based attacks, we are substantially reducing the search space and the false alarm rate, and the benefits we gain in terms of operational simplicity and reduced false alarm rate greatly outweigh the negligible decrease in the detection sensitivity.

The last potential pitfall, is related to the ability of the monitoring infrastructure to sustain data collection during attacks. While collecting low-volume SNMP feeds is not a serious overhead, collecting flow records at the customer egresses and transporting them back to a centralized processing engine is clearly infeasible during volume floods, since the access link is already overloaded, and reporting large volumes of flow records can only worsen the access link congestion. Large providers typically deploy flow collectors at core network elements, which are usually well-provisioned, and they can subsequently map the flow records to the appropriate egresses using routing and address space information. Thus, there will be be no perceivable reduction in the data quality and collection capabilities during attacks.

4 Implementation

We implemented LADS as an *off-line* DDoS detection system within a tier-1 ISP. The described implementation works on the real data of the tier-1 ISP and is only classified as *off-line* in that the data provided to the system might be substantially delayed and, therefore, is not suitable for generating alarms in real time. We are actively working on deploying the system in an *on-line* environment in which real-time data feeds are available, and our performance evaluation Section 6.1 indicate that our design and implementation will be adept to the task of online monitoring.

In our implementation we use SNMP data to compute our low level triggers and Netflow data to decide if the observed traffic indicates a potential attack. We describe each of the components in the remainder of this section.

4.1 Lightweight Anomaly Detection: SNMP

The first stage of the detection system, uses SNMP link utilization data to report volume anomaly triggers. Figure 3 provides a conceptual overview of our SNMP anomaly detection module. Specifically, we are interested in flow anomalies on egress interfaces that are associated with customer networks, since volume based DDoS attacks will be most visible at the egress links of the customers under attack. The SNMP data is being collected on an on-going basis and contains CPU and link loads (in terms of octet and packet) counts. Since most DDoS attacks use small packets [18] we use the egress

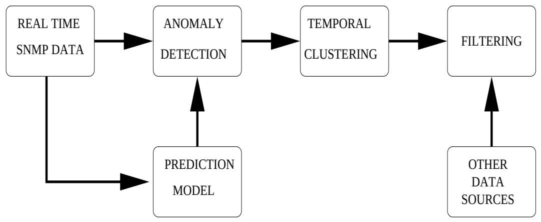

Figure 3: Overview of SNMP Anomaly Detection

```
TimeDomainModeling($TS, W, N$)
    ▷ TS is the training set
    ▷ W is the number of weeks
    ▷ N is the number of data points per week
1   for $i \leftarrow 1$ To N
        do
2           $P(i) \leftarrow$ mean($TS(1 : W, i)$)
3           $P' \leftarrow$ denoise($P$)
4           $V(i) \leftarrow$ VarianceModel($TS, P', W, N$)
5   return P',V
```

Figure 4: Time Domain Modeling Procedure

packet counts (i.e., with reference to Figure 1, the C-PE interface towards the customer) to detect volume anomalies.

To keep the operational, storage and computation resources low we devised a simple trigger algorithm with good performance (as we will show in Section 6). Conceptually, the trigger algorithm uses a prediction model which indicates an expected mean and an expected variance for the traffic time series, and assigns a deviation score, in terms of the number of standard deviations away from the mean that the given observation is found to be. Borrowing some formal notation from [24] one can think of the traffic time series, denoted by $T(t)$ as being composed of three components, $T(t) = P(t) + V(t) + A(t)$, where $P(t)$ represents the predicted mean traffic rate, $V(t)$ represents the stochastic noise that one expects for the traffic, and $A(t)$ is the anomaly component.

```
AnomalyDetection($T, TS, W, N$)
    ▷ T is the new time series
    ▷ TS is the historical time series
    ▷ W is the number of weeks for building model
    ▷ N is the number of data points per week
1   $(P, V) \leftarrow$ TimeDomainModeling($TS, W, N$)
2   for $i \leftarrow 1$ To N
        do
3           $D(i) \leftarrow (T(i) - P(i))/V(i)$
4   Do TemporalCluster($D, \alpha_{trigger}, \alpha_{add}, keepalive$)
5   Use filtering rules on clustered alarms
```

Figure 5: Outline of overall SNMP anomaly detection

Our algorithm, depicted in Figure 4, works as follows. For each customer interface, we take the last k weeks of data. We build an empirical mean-variance model using these by simple point-wise averaging, assuming a basic periodicity of one week. For example, for estimating the expected traffic for the 5 minute interval *Fri 9:00-9:05 am*, we take the k past Fridays and average of the observations for this time duration. Next, as the training data might contain DDoS attacks and is finite in size we perform a de-noising step using a Fourier transform from which we pick the top 50 energy coefficients. In the final step, for each point per week (e.g., Fri 9:00-9:05, Mon 21:00-21:05), the algorithm determines the variance over the last k observed data points with respect to the de-noised mean model.

The implicit assumption in the method is that the basic periodicity of the traffic data is one week, which has been used in other traffic analysis on such datasets [24].

The real-time anomaly detection is then performed using the algorithms described in Figure 5 and Figure 6. At a high level we use the estimated mean and the deviation time series to obtain deviation scores for the new observations. We use a natural definition of the deviation as $D(t) = (T(t) - P(t))/V(t)$, which represents the number of standard deviations away from the prediction that the observed data point is. Once the deviation scores have been computed, we perform a simple heuristic temporal clustering procedure to report anomalous incidents to the flow collector and analyzer. Temporal clustering can reduce the load on the collection mechanism by reducing the number of queries that we issue to the collector. Such a load reduction is indeed significant, as many attacks last quite long [18]. The clustering method operates based on two pre-defined deviation score thresholds, the event trigger threshold and the event extension threshold, as well as a keep alive time. The actual clustering procedure is simple: it tries to extend the current active event, if the new observation has a deviation score that exceeds the event extension threshold α_{add}, within a time duration of $keepalive$, since the start of the event. If there is no active ongoing event, it creates a new event if the observed deviation score is higher than the event trigger threshold.

After detecting the SNMP anomalies, we perform additional filtering steps to allow the operators to remove known or uninteresting anomalies. In particular we perform the following steps.

- Absolute volume threshold – This threshold is used to remove all SNMP alarms which do not have an average bandwidth of more than a pre-defined threshold. This allows the operator to specify a minimum attack rate of interest, to reduce the overall workload for the flow collector.
- SNMP measurement anomalies – We remove anomalies in the SNMP data caused by router resets and SNMP implementation bugs. In particular we remove the first SNMP counters after a reset as well as measurements which indicate a bandwidth utilization of more then the physical bandwidth [3].

At the end of the SNMP anomaly stage, we receive a set of alarms, specified by the egress interface on which the anomaly was observed, and the start and timestamps of the anomaly incident. These alarms are then used to trigger Netflow collectors for detailed investigation.

```
TEMPORALCLUSTER($D, \alpha_{trigger}, \alpha_{add}, keepalive$)
    ▷ D is the deviation score time series
    ▷ $\alpha_{trigger}$ is trigger deviation score threshold
    ▷ $\alpha_{add}$ is the score threshold for extending an event
    ▷ $keepalive$ is the time for which an event is active
    ▷ The output is the set of alarms
 1  for $i \leftarrow 1$ To N
    ▷ Flag keeps track whether there is an ongoing alarm
    ▷ T is the current time
    ▷ ST keeps track of event times
        do
 2          if ($Flag = TRUE$)
              then
 3                if $D(i) \geq \alpha_{add}$
 4                  then Extend the event
                  ▷ E is the current event
 5                $\Delta T \leftarrow T - ST(E)$
 6                if $\Delta T \geq keepalive$
 7                  then $Flag \leftarrow FALSE$
              else
 8                if $D(i) \geq \alpha_{trigger}$
                    then
 9                      Create a new event E'
10                      $Flag \leftarrow TRUE$
11                      $ST(E') \leftarrow T$
12  Return alarms with durations and scores
```

Figure 6: Temporal clustering to cluster deviation scores

4.2 Focused Anomaly Detection: Netflow

The second stage of our DDoS detection system analysis the Netflow data as shown in Figure 7. At a high level we perform the following steps for each SNMP alarm:

- Collect all Netflow records for the egress interface indicated by the first stage trigger.
- Build three Netflow record sets containing
  - Records with the TCP SYN flag set (SYN set)
  - Records with the TCP RST flag set (RST set)
  - Records for the ICMP protocol (ICMP set)

[3]Even though the number of such events is low, they do occur daily on a large network.

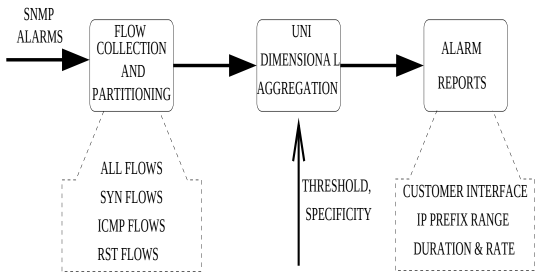

Figure 7: Incident diagnosis using flow data

  – All flow records

- For each set report the traffic volumes for all destination prefixes with a prefix length larger than a /28, use the uni-dimensional clustering algorithm described in Figure 8. Generate a final bandwidth attack alarm if the all-set has a /28 which carries more traffic then the configurable Bandwidth Attack Threshold. Generate final SYN/ICMP/RST alarm if the corresponding SYN/ICMP/RST flow data indicate a IP prefix range which carries more traffic than the configurable SYN/ICMP/RST Threshold. Instead of using a fixed rate threshold across all alarms, we use a duration-adaptive rate threshold, that attempts to balance the sensitivity between high intensity low duration attacks, and relatively lower intensity but higher duration attacks. This can be achieved by using a simple depreciation approach, so that the rate threshold is a monotonically decreasing function of the alarm duration. Our current implementation uses a geometrically decreasing depreciation, where the average rate for longer duration events will be generated according to the following formula $Rate(Duration) = Rate(BaseDuration) * DecreaseFactor^{Duration/BaseDuration}$, where the $BaseDuration$ is 300 seconds, and the $DecreaseFactor$ is set to 0.95.

The procedure for uni-dimensional clustering is described in Figure 8. There are two steps of the uni-dimensional clustering: Aggregation and Reporting. The aggregation step simply counts the total traffic volume received by each distinct destination prefix, larger than a minimum prefix-range size, denoted by $MinPrefix$. Since we are interested in DDoS attacks on customer egress links, we can afford to perform the traffic aggregation on smaller prefix ranges, than would be the case for more general purpose traffic analysis applications [9, 33, 30]. Thus the compute and memory overhead during the aggregation phase is upper-bounded by the size of the prefix range size we are interested in. For example, we are only interested in the traffic intended for prefixes larger than a /28, which can be potential attack targets. The next step is the *Reporting* stage, which uses the aggregated counters to decide whether to report the particular prefix range as being a potential attack target. The reporting step we use shares conceptual similarity with the work of Estan et al. [9] and Singh et al. [33], to generate traffic summaries indicating heavy-hitters. Intuitively, we generate reports on larger prefixes, if they carry substantially more traffic than a previously reported smaller prefix range, and if they are above the absolute volume threshold. We scale the absolute volume threshold according to the size of the prefix range by a multiplicative *Specificity* parameter that determines the scaling factor. We chose this approach due to its simplicity and we observe that the diagnostic capability provided by our approach is sufficient for detecting DDoS attacks, and generating alarm reports comparable to commercial DDoS solutions (Section 6.4). We also found in our evaluation that this approach is computationally efficient, in terms of memory and processing time, which make it a viable alternative for near real time analysis.

5 Experimental Setup

To evaluate our LADS implementation we collected SNMP and Netflow data for a subset of the access interfaces of a tier 1 ISP ranging from T1 to OC48 speeds. We describe this data collection next followed by a description of the LADS parameter settings we used.

5.1 Data Description

For our LADS evaluation we collected SNMP and Netflow data for a 22000 subset of interfaces within a large ISP. To allow our evaluation to be repeatable during development we archived all relevant data for an 11 day period in August 2005 with the exception of the SNMP data used which was archived for a period in excess of 12 months.

In particular we collected the following datasets:

SNMP Our SNMP dataset contains the in and out byte and packet counts for all interfaces considered. Currently we only utilize the packet out counts which count the packets from the ISP to the customer.

NetFlow Data The NetFlow data contains sampled NetFlow records covering the entire backbone network. The records are based on 1:500 packet sampled data. The sampling is performed on the router and the records are subsequently smart sampled to reduce the volume. In smart sampling, flow records representing total bytes greater than a threshold z of 20MBytes are always sampled, while smaller records are sampled with a probability proportional to their size. Appropriate renormalization of the reported bytes yield unbiased estimates of the

```
UNIDIMENDIONALCLUSTERING($MinPrefix, Threshold, Specificity$)
    ▷ MinPrefix is the minimum prefix length – Set to 28, MaxPrefix is the maximum IP prefix length (32 for IPv4)
    ▷ Threshold is given in terms of an attack rate, Specificity is used for compressing the report – Set to 1.5
1   Aggregation: Read flow records and update traffic counts for each unique prefix between MinPrefix and MaxPrefix
2   Reporting: for $i \leftarrow MaxPrefix$ DownTo $MinPrefix$
        do
3           for each Prefix $P$ of prefix-length $i$
                do
                    ▷ We use the IP/Prefix notation, $P/\{i\}$ refers to prefix $P$ with a prefixmask of length $i$
4                   $AbsoluteThreshold \leftarrow Specificity^{(MaxPrefix-i)} \times Threshold$
5                   if $i \neq MaxPrefix$
6                      then $CompressThreshold \leftarrow Specificity \times PredictedVol(P/\{i\})$
7                      else $CompressThreshold \leftarrow 0$
8                   $ReportThreshold \leftarrow$ MAX$(AbsoluteThreshold, CompressThreshold)$
9                   if $Volume(P) > ReportThreshold$
10                     then Report alarm on prefix $P/\{i\}$ with rate Volume(P)
11                  $PredictedVol(P/\{i-1\}) \leftarrow$ MAX$(PredictedVol(P/\{i-1\}), Volume(P))$
```

Figure 8: Procedure for uni-dimensional prefix aggregation and generating compressed reports

traffic bytes prior to sampling [8]. In the resulting data set each record represent on average at least 20MByte of data. After collecting the records we also annotate each record with its customer egress interface (if it was not collected on the egress router) using route simulation and tag records which could have been observed twice within the collection infrastructure to avoid double counting of netflow records.

Commercial-alarms The tier-1 ISP currently has a commercial DDoS detection system deployed at key locations within its network. We collected the high priority DDoS alarms from this commercial DDoS detection system. The alarms where combined into attack records if we found multiple alarms for the same target with an idle time of less then 15 minutes in between alarms. Even though we are not aware of the detailed algorithms used within this product our operational experience indicates that the system detects most large DDoS attacks while generating a manageable amount of high priority alarms. The system is deployed in a substantial fraction of the core of the ISP at high speed interfaces and, therefore, only analyzes aggregate customer traffic. Again we used route simulation to determine the egress interface of the attack traffic. This dataset is used to compare our approach to a commercially available system which requires additional hardware deployment. In an ideal scenario, we would like to evaluate the false positive and false negative rates of our implementation against some *absolute ground truth*. However, we are not aware of any system which can generate such ground truth at the scale that are of interest for our system. Hence, we use the commercial detection system as a basis for comparison with our implementation even though we are aware that due to its deployment locations and configuration (we only collect high priority alarms) the commercial system might not detect some of the DDoS attacks which are detectable with our system.

5.2 System Configuration

In terms of the specifics of our implementation, our approach requires a number of configurable parameters which we set to the following values:

SNMP training period We set the training period for model building for the SNMP anomaly detection to be 5 weeks.

Absolute Volume Threshold The absolute volume threshold provides a lower bound on DDoS attacks we detect in the SNMP data. We set this value to 250kbps which considering that the smallest link size in the Tier-1 ISP's network is a T1 (1.5Mbps) allows us to detect any sizable attack on any interface under consideration.

Event Score Threshold The threshold on the deviation score which triggers an SNMP based alarm. We evaluate the sensitivity and overhead for different threshold values in Section 6.2.2. For our evaluation we use an Event Score Threshold of 5.

Temporal Clustering Parameters The temporal clustering procedure uses an Event Extension Threshold and a KeepAlive duration value, for deciding on combining SNMP alarms. We set the event extension threshold to be half the Event Score Threshold, and the KeepAlive duration to be 15 minutes.

Bandwidth Attack Threshold This threshold is applied to determine within the netflow data if a particular incident should be reported as a potential DDoS attack, if none of the other DDoS related signatures (e.g. high volumes of SYN, ICMP, or RST packets) are present. We set this threshold to a high-intensity threshold of 26 Mbps [4], targeted at a single /32 behind a customer interface. The rate for alarms of longer duration will be lower due to the rate depreciation described in Section 4.2. The thresholds for larger prefixes (upto /28) are scaled according to the algorithm described in Figure 8.

SYN/ICMP/RST Threshold This threshold is applied to determine within the netflow data if a particular incident could be considered a SYN, ICMP or RST attack. Currently we set this rate to a high intensity rate of 2.6 Mbps[5], averaged over a 300 second interval. Again, we use a similar rate depreciation function for longer duration alarms.

6 Experimental Results

In this section we evaluate LADS. In Section 5 we described the data that we used in our evaluation as well as the specific parameters chosen for the different parts of our system. We first study our system performance in Section 6.1, followed by an evaluation of the SNMP based trigger phase in Section 6.2, before analyzing the incidents generated by our system in Section 6.3.

6.1 Performance

The data was collected using an existing SNMP and Netflow data collection infrastructure. The SNMP data is being collected by a commercial of the shelf SNMP collection tool which seems to scale easily to large networks. The Netflow collection system on the other hand was specifically build for this large ISP and is described in more detail in [8]. Currently this infrastructures monitors in excess of one petabyte of data each day.

Using these existing data sources we implemented our data extraction using a combination of flat files and an inhouse database system. All our data-extraction, and analysis modules were implemented in Perl, with very few performance optimizations. The model-building phase uses additional MATLAB scripts for performing the denoising and cleaning operations described in Section 4.

The model-building phase which uses one month of data per interface to get a mean-variance model for the anomaly detection, takes roughly 26 hours to perform the data extraction, de-noising, and model extraction for all the 22000 interfaces. This is not a concern since this part of the analysis can be performed offline, and the overhead is acceptable as it is not on the critical path for near real-time attack detection.

We generated the SNMP and final alarms for our entire 11 day period on a heavily shared multi-processor 900MhZ SUN Ultra. The anomaly detection stage was parallelized using 6 processes, and it takes roughly 11.2 seconds to report the deviation scores, for each 5 minute interval, across the 22000 interfaces used in our evaluation. The biggest bottleneck for our performance is the extraction of flow records for each reported alarm (even after the reduction due to the triggers), the main reason being that (a) all flow data is currently compressed to meet the storage constraints, and (b) the flow data is collected on a per-collector basis and not indexed based on the egress interface. Even with these performance inhibitors, it takes around 212.5 seconds to map the flow data for each 5 minute interval, to the appropriate egress interfaces and collect the flow data that needs to be analyzed. We note that this time can be reduced significantly by indexing the data using appropriate database technology.

The last stage of our analysis does the uni-dimensional aggregation on the collected flow data, taking approximately 40 seconds for each 5 minute interval. Thus, for each 5 minute interval of data arriving at the processing engine, the total time that is needed to analyze the data and report the alarms is the sum of the time taken to generate deviation scores, the time taken to extract flow data for the triggered data, and the time taken to process the flow data, which is equal to $11.2 + 212.5 + 40 = 263.7 seconds$ to process each 5-minute interval. The resulting maximum latency with which we will report an alarm is, therefore, at most 263.7 seconds (or less than 5 minutes), implying that even with our current unoptimized implementation we can perform the near realtime analysis. On a more state of the art platform (900MhZ UltraSparcs are quit dated) , and with additional optimizations both in our implementation and the data indexing we expect to achieve substantially better performance.

6.2 SNMP-based Trigger Evaluation

We evaluate our SNMP based trigger implementation in three stages. First, we discuss the choice of our trigger algorithm, then we compare our trigger events against the commercial-alarms and finally we highlight the savings our triggered approach provides.

[4]Our implementation counts the total number of bytes and sets a base rate of 2000000 bytes every 300 seconds on the smart-sampled data, which roughly translates into a raw data rate of $\frac{2000000*500*8}{300} \approx 26Mbps$.

[5]Our actual implementation counts the number of distinct flows and sets a threshold of 5 flows every 600 seconds, which translates into an absolute data rate of $\frac{5*20MB*8}{300} \approx 2.6Mbps$.

6.2.1 Choice of Algorithm

In the context of our system we are looking for a model which is efficient, works on historical data only and detects anomalies early. Those requirements are motivated by the facts that we have to perform this analysis in real time on tens of thousand of times series to provide DDoS alarms within a reasonable timeframe. Our mean-variance based model provides these features, however, one interesting question is how our results compare to the results of more complicated algorithms.

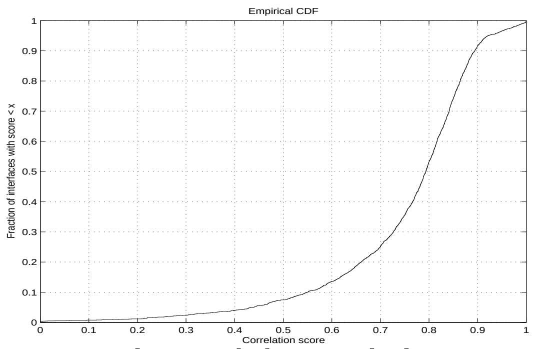

Figure 9: Correlation with the anomaly detection procedure in [24]

Figure 9 depicts the correlation of our trigger algorithm with the one proposed in [24]. The basic difference between these approaches lies in the assumption about the variance of the time-series. We use an empirical data-driven approach while Roughan et al. [24] assume that the stochastic component is of the form $\sqrt{a \times P_t}$, where a is a peakedness factor, and P_t is the periodic component of the time-series model (obtained using moving averages). Figure 9 shows a correlation score of greater than 0.7 between these two methods for more than 75% of all the 22000 interfaces selected for detection. Considering that different detection methods and models will always have different sensitivities and that there is no ground truth available to us, we conclude that in our problem domain the simple trigger model has similar properties to more complex models and is adequate to perform the trigger function.

6.2.2 Accuracy

Another interesting question is how frequently the SNMP based trigger would miss an attack in the commercial-alarm set. It is important to repeat here that we believe based on our operational experience that most such alarms are true and that the goal of the trigger phase of our system is to reduce the data to a manageable amount not to remove all false positives. Therefore, the interesting question is how many alarms present in the commercial-alarm set do not trigger an SNMP alarm using our model.

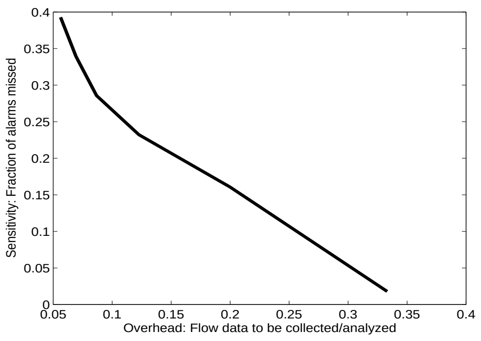

Figure 10: Tradeoff between sensitivity and scalability

Figure 10 depicts the tradeoff between the sensitivity of the triggers and the overall data complexity reduction the trigger can achieve. We define as the sensitivity of the trigger (for a particular deviation score threshold) the percentage of commercial-alarms which match an SNMP alarm for the threshold. The data complexity reduction achieved by the trigger can be calculated in terms of the flow data that will be collected after the trigger step and which needs to be further analyzed. Ideally, we would like to have a trigger that has perfect sensitivity (i.e., zero false negative rate), that can also produce very low collection and computation overhead.

It does appear that we can capture around 80-90% of all the commercial-alarms purely by looking at the SNMP alarms. As a tradeoff between sensitivity and data reduction we chose a 80% reduction rate (i.e., only 20% of the original flow data needs to be collected and analyzed in the second stage of our system). This results in an 85% overlap with the commercial-alarms and an anomaly detection threshold of 5 which we use for the remainder of this paper. We will further discuss in Section 6.4 the commercial-alarms not covered, which do not fit the profile of attacks we expect to detect. From a provider perspective the alarms that are of paramount importance are those affect the customers the most, and typically these are attacks which overload the egress link. If the misses occur on well-provisioned interfaces, this loss in sensitivity is not a serious concern.

Figure 11 shows the number of incidents per customer interface per day over the 11 day evaluation period before and after applying the Absolute Volume Threshold. It seems the Absolute Volume Threshold reduces the number of SNMP alarms on average by a factor of 6. Fortunately, in either case the number of alarms is quite low considering that these alarms are automatically processed by the second phase of our detection system.

6.3 Incident Analysis

Next we characterize the final alarm set our detection system generates after performing the netflow analysis de-

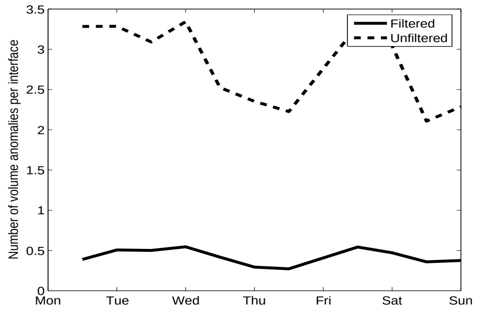

Figure 11: Number of alarms per interface per day

scribed in Section 4.2. As previously described each final alarm specifies a duration, an egress interface, destination IP-prefixes along with type of the alarm type (BW,SYN,RST,ICMP) and the bandwidth of the suspected DDoS event which is the information we used for the following graphs.

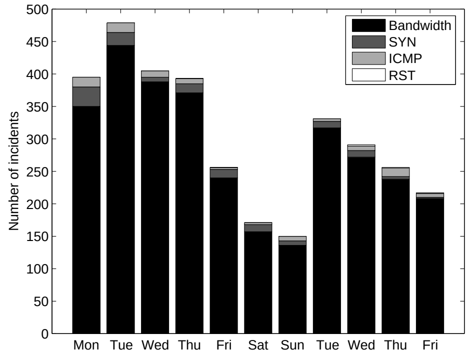

Figure 12: Number of reported incidents (at egress interface granularity) in 11 day period in Aug 2005

Figure 12 shows the number of these alarms during our evaluation time[6]. Here we consider incidents at the granularity of egress interfaces, i.e., concurrent attacks against multiple IP addresses on the same egress interface will be considered one incident. We generate approximately 15 incidents per hour which seems reasonable considering that we monitor 22000 customer interfaces and that this number of incidents could easily be handled by the security staff of a large network provider. We note that a large fraction of the incidents are reported as potential bandwidth attacks. One possible explanation could be that these incidents are bandwidth floods that are not necessarily DDoS attacks, and are potentially false positives from the perspective of attack detection. In the absence of any ground truth regarding these incidents, we believe there is utility in bringing such incidents to the notice of network operators – since there was a definite volume anomaly on the interface, and these hosts are receiving a high data rate during this volume anomaly.

[6]Due to data collection issues we miss the data of the second Monday in our evaluation time

The number of distinct IP addresses involved in an attack might be a more reliable indicator of the work involved analyzing these alarms than the number of egress interfaces. If we split all alarms which target multiple IP addresses at a single egress interface into multiple alarms, we only see a slightly higher alarm rate (around 18 per hour) . That is not surprising considering that most (76%) of the incidents involve only one IP address.

Another observation is that in around 19% of the cases we get repeated alarms for the same IP address range within the same day. These alarms would most likely only require one investigation attempt by the network operator. Therefore, we believe that the above alarm rates are actually an upper bound of the number of trouble tickets network operators need to process.

6.4 Comparison with Commercial DDoS Detection System

In this section we compare our final LADS alarms against the commercial-alarms. Since the commercial DDoS detection system only covers key locations and we only receive high level alarms from this system we would expect that our final alarm set contains substantially more attacks then the commercial-alarms. This is indeed the case in that the commercial system only reports 86 such high level alarms involving the interfaces we analyzed, whereas our system generated a total of 3314 final alarms, over the 11 day period. Therefore, we mainly use this data set to determine the false negative rate of our approach since we expect that we should have detected most alarms which are also detected by the commercial system. We are currently investigating techniques for using historical uni-dimensional cluster reports (on a per-prefix basis) to capture repeating bandwidth events which could cause false positives in the current set of LADS bandwidth alarms. Also, routing information could be used to restrict the set of alarms generated by our system to traffic covered by the commercial system, and so compare performance with the same traffic set.

Figure 13 presents a breakdown of the comparison of our final alarms versus the commercial-alarms. The breakdown uses the following categories to classify the 86 alarms of the commercial system.

Successes Between the LADS alarms and the commercial-alarms the interface matches, the IP prefix alarmed matches, and the durations of the reported alarms overlap.

Found early incidents Between the LADS alarms and the commercial-alarms the interface matches, the ip address alarmed matches, but we find the alarm slightly earlier than what is reported.

Found late incidents Between the LADS alarms and the commercial-alarms the interface matches, the ip address alarmed matches, but we find the alarm slightly later than what is reported.

Threshold misses The interface matches, we have an SNMP volume anomaly, and we have flow data for the incident that indicates a large number of flows to the IP, but we missed alarming on the IP address during the second phase of detection.

Anomaly detection misses This commercial-alarm did not generate an SNMP alarm on the reported interface, i.e., the deviation score for the corresponding time in the SNMP dataset is less than our SNMP alarm threshold (which is set to 5 for our evaluation).

Potential commercial-alarm false positive The interface information and the anomaly match between our SNMP alarm and the commercial-alarm, however, we find little or no flow data for the corresponding attack target reported by the alarms.

The false negative rate of our system compared to the commercial DDoS detection system is essentially the sum of the anomaly misses, and threshold misses. Manual analysis of the anomaly detection misses indicates that all 7 SNMP anomaly misses are caused by relatively small attacks on OC48 interfaces (2.48Gbit/sec). They did not saturate the customer interface and therefore are not in the category of DDoS attacks we want to detect. The number of threshold misses on our system is low - just 1 incidents out of 86 incidents are dropped due to the threshold settings. Therefore we conclude that the overall false negative rate of our system, compared to a commercial DDoS detection system, is 1 out of 80, or 1.25%.

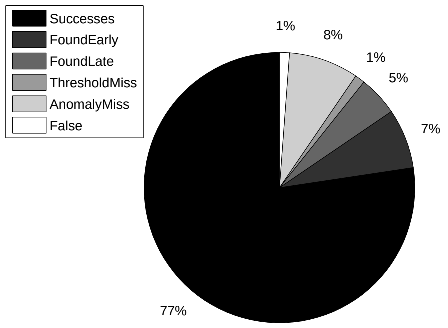

Figure 13: Comparison with the proprietary system

To compare the type of attacks that are found in the overlap between the vendor solution and our system – we give a breakdown of the 4 types of incidents *Bandwidth, SYN, ICMP, RST* in Figure 14. Interestingly, the largest portion of the reported incidents which overlap are SYN floods.

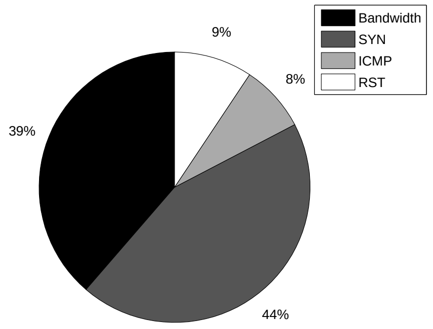

Figure 14: Breakdown of overlapping incidents

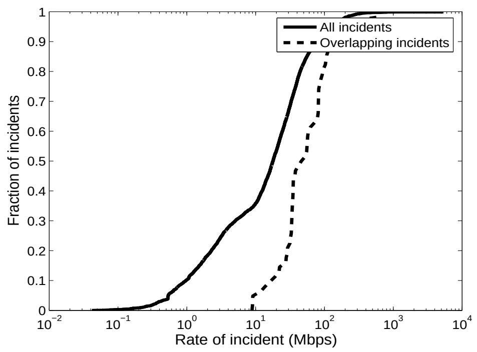

Figure 15: Rates of potential attack incidents

Another dimension of comparing the final LADS alarms with the commercial solution is depicted in Figure 15 which shows the average bitrate according to our netflow records of the DDoS events alarmed on by our system and the commercial DDoS detection system. The overlapping incidents appear to have a minimum rate of 10 Mbps, which is most likely due to the fact that we only had access to the high priority alarms of the commercial DDoS detection system. Interestingly, this makes the high level alerts of this system unsuitable for detecting DDoS attacks against small customer links. The system therefore ranks attacks as high level alerts not by customer impact (even a small attacked customer link has a lot of customer impact for the customer using that link) but by the overall attack size. This is of course less desirable, if the goal is to protect customers which subscribe using various line rates. For 40% of the final LADS alarms we find a reported bandwidth which is smaller than 10Mbps [7]. Further investigation reveals that more than 70% of these low volume alarms are in fact caused by volume floods against low speed links.

[7]The rates for alarms of duration longer than 300 seconds will be lower than the high intensity thresholds of 26 Mbps for the bandwidth attacks, and 2.6 Mbps for SYN/RST/ICMP attacks, due to the rate depreciation we discussed earlier.

7 Conclusions

We presented the design of a general triggered framework for scalable threat analysis, and a specific implementation based on SNMP and Netflow feeds derived from a large Internet provider. Our evaluations and experience with large networking datasets demonstrate the imminent need for such an approach. Our results indicate that the particular system we built is adequate for the task of detecting attacks at scale, and doing so with significant reduction in operational complexity and computational cost. Of particular practical significance is the fact that our system uses data feeds that are readily available to most providers.

There are several interesting directions for future work. Our evaluations demonstrate that our design is sufficient, but we believe there is scope for improving the individual components (anomaly detection, flow analysis). We are also investigating other ways in which we can confirm the validity of the alarms generated by our system, including those mentioned in Section 6.4. Finally, we are currently pursuing the implementation of more real time feeds to our system to allow us to use it in an ongoing basis as a online threat detection mechanism.

References

[1] Cisco guard. `http://www.cisco.com/en/US/products/ps5888/`.

[2] ADKINS, D., LAKSHMINARAYANAN, K., PERRIG, A., AND STOICA, I. Taming IP Packet Flooding Attacks. In *ACM SIGCOMM HotNets II* (2003).

[3] Arbor Networks. `http://www.arbor.com`.

[4] BARFORD, P., KLINE, J., PLONKA, D., AND RON, A. A Signal Analysis of Network Traffic Anomalies. In *Proc. of ACM/USENIX IMW* (2002).

[5] BRUTLAG, J. D. Aberrant Behavior Detection in Time Series for Network Monitoring. In *Proc. of USENIX LISA* (2000).

[6] BURCH, H., AND CHESWICK, B. Tracing Anonymous Packets to Their Approximate Source. In *USENIX LISA* (2000).

[7] COOKE, E., JAHANIAN, F., AND MCPHERSON, D. The zombie roundup: Understanding, detecting, and disrupting botnets. SRUTI Workshop, 2005.

[8] DUFFIELD, N., LUND, C., AND THORUP, M. Learn more, sample less: control of volume and variance in network measurement. *IEEE Transactions in Information Theory 51*, 5 (2005), 1756–1775.

[9] ESTAN, C., SAVAGE, S., AND VARGHESE, G. Automatically Inferring Patterns of Resource Consumption in Network Traffic. In *Proc. of ACM SIGCOMM* (2003).

[10] GIL, T., AND POLETTO, M. Multops: a data-structure for bandwidth attack detection. In *Proc. of USENIX Security Symposium* (2001).

[11] KANDULA, S., KATABI, D., JACOB, M., AND BERGER, A. Botz-4-sale: Surviving organized ddos attacks that mimic flash crowds. In *Proc of ACM/USENIX NSDI* (2005).

[12] KEROMYTIS, A., MISRA, V., AND RUBENSTEIN, D. Sos: Secure overlay services. In *Proc. of ACM SIGCOMM* (2002).

[13] KOMPELLA, R. R., SINGH, S., AND VARGHESE, G. On Scalable Attack Detection in the Network. In *Proc. of Second ACM/USENIX IMC* (2004).

[14] KRISHNAMURTHY, B., MADHYASTHA, H. V., AND SPATSCHECK, O. ATMEN: a triggered network measurement infrastructure. In *International World Wide Web Conference* (2005).

[15] LAKHINA, A., CROVELLA, M., AND DIOT, C. Diagnosing network-wide traffic anomalies. In *Proc. of ACM SIGCOMM* (2004).

[16] LAKHINA, A., CROVELLA, M., AND DIOT, C. Mining anomalies using traffic feature distributions. In *In ACM SIGCOMM* (2005).

[17] MAHAJAN, R., BELLOVIN, S., FLOYD, S., IOANNIDIS, J., PAXSON, V., AND SCOTT, P. Controlling high bandwidth aggregates in the network. *ACM SIGCOMM CCR 32* (2002).

[18] MAO Z., SPATSCHECK O., VAN DER MERWE J. AND VASUDEVA R. Analyzing large ddos attacks using multiple data sources. Under submission., Oct 2005.

[19] Mazu Networks. `http://www.mazu.com`.

[20] MIRKOVIC, J., AND REIHER, P. A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms. *ACM SIGCOMM CCR 34* (2004).

[21] MOORE, D., VOELKER, G. M., AND SAVAGE, S. Inferring Internet Denial-of-Service activity. In *USENIX Security Symposium* (2001).

[22] MOREIN, W. G., STAVROU, A., COOK, D. L., KEROMYTIS, A., MISRA, V., AND RUBENSTEIN, D. Using graphic turing tests to counter automated ddos attacks against web servers. In *Proc. of ISOC NDSS* (2005).

[23] PERRIG, A., AND SONG, D. Advanced and authenticatd marking schemes for ip traceback. In *Proc. of IEEE INFOCOM* (2001).

[24] ROUGHAN, M., GREENBERG, A., KALMANEK, C., RUMSEWICZ, M., YATES, J., AND ZHANG, Y. Experience in measuring internet backbone traffic variability: Models, metrics, measurements and meaning. In *Proc. of International Teletraffic Congress (ITC)* (2003).

[25] SANDERS, T. Cops smash 100,000 node botnet. http://www.vnunet.com/2143475, Oct 2005.

[26] SAVAGE, S., WETHERALL, D., KARLIN, A., AND ANDERSON, T. Practical Network Support for IP Traceback. In *ACM SIGCOMM* (2000).

[27] SNOEREN, A. C., PARTRIDGE, C., SANCHEZ, L. A., JONES, C. E., TCHAKOUNTIO, F., KENT, S. T., AND STRAYER, W. T. Hash-Based IP Traceback . In *ACM SIGCOMM* (2001).

[28] STONE, R. Centertrack: An IP Overlay Network for Tracking DoS Floods. In *Proc. of USENIX Security Symposium* (2000).

[29] THE HONEYNET PROJECT. Know your enemy: Tracking botnets. `http://www.honeynet.org/papers/bots`.

[30] XU, K., ZHANG, Z.-L., AND BHATTACHARYA, S. Profiling internet backbone traffic: Behavior models and applications. In *In ACM SIGCOMM* (2005).

[31] YANG, X., WETHERALL, D., AND ANDERSON, T. A dos-limiting network architecture. In *Proc. of ACM SIGCOMM* (2005).

[32] ZHANG, Y., GE, Z., GREENBERG, A., AND ROUGHAN, M. Network anomography. In *In ACM/USENIX IMC* (2005).

[33] ZHANG, Y., SINGH, S., SEN, S., DUFFIELD, N., AND LUND, C. Online detection of hierarchical heavy hitters: algorithms, evaluation, and applications. In *Proc. of Second ACM/USENIX IMC* (2004).