Detecting DDoS Attacks on ISP Networks

Aditya Akella Ashwin Bharambe Mike Reiter Srinivasan Seshan

Carnegie Mellon University

Abstract

Most past solutions for detecting denial of service attacks (and identifying the perpetrators) have targeted end-node victims. However, little attention has been given to this problem from an ISP perspective. This paper explores the key challenges involved in helping an ISP network detect attacks on itself or attacks on external sites which use the ISP network. We propose a detection mechanism where each router detects traffic anamolies using profiles of normal traffic constructed using stream sampling algorithms. In addition, an ISP's routers exchange information with each other to increase confidence in their detection decisions. Our initial results show that individual router profiles capture key characteristics of the traffic effectively and help identify anomalies with low false positive and false negative rates. We believe that profile construction can be extremely efficient, supporting even multi-gigabit speeds. We also believe that incremental deployment of such techniques is possible, although it may signficantly impact the effectiveness of the distributed reinforced decision making.

1 Introduction

Distributed Denial of Service (DDoS) attacks have become an increasingly frequent disturbance in today's Internet. Many recent research efforts have explored designing mechanisms for detecting such attacks and identifying the perpetrators. However, all these solutions are aimed at aiding end-node victims under attack. In this work, we look at the problem from the point of view of an Internet Service Provider (*ISP*). Specifically, we design mechanisms that allow ISPs to quickly and efficiently answer the following questions: (1) Is the ISP backbone itself under a DDoS attack? (2) Is the ISP network carrying much "useless" [1] traffic? (3) Which traffic is malicious and what should be done to such traffic?

In today's BGP-driven Internet, large ASes peer with other ASes at multiple PoPs (Points of Presence). If a packet's destination is not within itself, an AS hands over the packet to other ASes as soon as possible. This *hot potato* routing may not use the shortest route to the destination. Due to these factors, packets going to the same destination can traverse diverse and disjoint paths through an AS. This "dispersion" makes it hard to detect DDoS traffic at any single point, necessitating a distributed approach to the problem.

Our approach to this problem relies on routers within the ISP identifying traffic pattern violations themselves. This is achieved by building traffic *profiles* using stream sampling algorithms which have an extremely small memory footprint. By sampling over relatively long time windows, *normal* traffic profiles are created while *current* traffic profiles are constructed by using smaller time windows. Whenever the current profile does not corroborate with the normal one, a router becomes *suspicious*.

The key challenge of this approach is making it robust. Can we ensure that our profiles are detailed enough that malicious attackers cannot disguise their attack traffic as normal traffic? How do we avoid detecting normal variations in traffic patterns (including unusual flash crowds) as attacks? Finally, can these profiles be collected efficiently without creating new opportunities for attack?

Our initial results using stream-sampling schemes to build profiles show that it is possible to: 1) profile normal traffic reasonably accurately, 2) identify anomalies with low false-positive and false-negative rates (locally, at the router), 3) consume low per-packet computation and memory even at very high router speeds.

However, if we rely only on this mechanism, small traffic perturbations within the ISP or in the traffic will trigger many false positives. We believe that this approach can be made more robust by having routers communicate their suspicions to other routers in the backbone. Routers aggregate the suspicions received from all other routers before deciding whether a certain traffic aggregate belongs to an attack or not. The only way an attacker can circumvent this mechanism is to successfully guise her traffic so as to fit the normal profile at a large number of routers. If the attacker just takes very few paths through the ISP network but still sends a large amount of traffic, she will be easily caught. By choosing the right set of statistics to constitute the normal profiles, we can ensure that the former approach of mimicking many profiles becomes extremely hard.

We describe an example of this approach in greater detail in Section 2 and present a brief discussion in Section 3.

2 Profile-Based DDoS Detection

It is clearly infeasible to keep traffic statistics for every single destination at a backbone router. Here, we consider a traffic profile that can be collected with little overhead and should be able to detect most intruders. At any point of time, only high-traffic destinations need be considered since those exactly are the ones which are likely to be under attack. Hence, each router keeps track of destinations whose traffic occupies greater than a fraction θ of the capacity C of the outgoing link using a *sample-and-hold* algorithm [2]. We call these destinations *popular* and destinations not in this list as *unpopular*.

Traffic profiles at each router are basically a set of *fingerprints* F_i of the traffic to popular destinations. An effective choice of such fingerprints is the key to characterizing traffic streams. However, computing arbitrary fingerprints might require excessive memory and/or computation. We have identified several fingerprints which can be efficiently computed using stream sampling algorithms. Some of them are:

- The total number of bytes to the destination, τ.

[1] Attack traffic aimed at a certain destination that is ultimately dropped at the destination.

- For various value of p, the number of /p prefixes sourcing traffic to the destination. The motivation is that this set of fingerprints characterizes source-subnet distribution and would catch random source spoofing by an attacker.
- An approximation to the flow-length distribution of traffic to the destination. We sample specific points on the flow-length distribution by keeping track of the number of source IP addresses that send more than θ_k fraction of the total traffic to the destination, for various values of θ_k.

We use *sample-and-hold* [2] and *zeroeth moment (F_0)* computation [4, 1] algorithms for computing these fingerprints. Each statistic is computed by sampling over a small interval of time, about a minute. To reflect the typical day-of-week and hour-of-day traffic patterns, routers construct per-hour, per-weekday *normal* traffic profiles by averaging the statistics over hourly periods.

Algorithm at Each Router. With these statistics in hand, each router R uses the following algorithm for a popular destination:

1. Let τ_b be the number of bytes to the destination in the base-profile and τ be the same statistic in current sampling interval. If $\tau > \tau_b + \theta C$, continue to next step. Otherwise, stop.
2. For each fingerprint F_i, let v_i denote the value computed in the current sampling interval. Let μ_i and σ_i denote the mean and standard deviation values for this fingerprint.[2] If $|v_i - \mu_i| > z\sigma_i$, then $flag(F_i) = 1$, else $flag(F_i) = 0$. $z > 0$ is a parameter to the algorithm.
3. Let *conf* denote the confidence with which the router R suspects an attack. We set $conf = \sum_i \delta(F_i) * flag(F_i)$. δ assigns "weights" to a fingerprint, depending on the extent to which that fingerprint contributes to errors (false-positive or negatives): $\delta(F_i) \propto \frac{1}{err(F_i)}$ where $err(F_i)$ is the sum of the false positive and negative rates for F_i. The ISP can perform measurements to determine the appropriate δ to configure routers with.

For each packet, we need to perform the two operations: run *sample-and-hold* [2] to probabilistically sample only large-volume destinations requiring one hash-table lookup or update and one byte operation; and if the packet was sampled, update statistics for the corresponding destination. For each *counting* statistic, we need to apply the algorithm in [4] involving one hash-table lookup or update and 3 byte operations. If we assume that all the memory operations are performed in SRAM, profile computation consumes few CPU cycles and can support very high data rates.

We need a slightly different mechanism for destinations which are usually unpopular but suddenly become popular. Since it is infeasible to keep in-core traffic statistics for such destinations, we store their base profiles on disk. This can be done by computing statistics for randomly sampled packets over time. When a router finds that $\tau > \theta C$ for a destination *not* in its popular list, it pages in the corresponding base profile from disk. If no such profile exists, all fingerprints are *flagged*. Otherwise, it computes flags and *conf* as stated above.

Distributed Detection. For each destination with $conf > 0$, each router sends the $(conf, dest)$ pair to its neighbors. On receiving such a message, the neighbors discard duplicates, compute the aggregate, $Aggr$ of the $conf$ values received per destination and forward non-duplicates along to their neighbors. If, for any destination, $Aggr$ exceeds a pre-defined threshold, the router concludes that the destination is under attack. This "consensus" stage helps reduce the errors in identification of attacks even further. These messages could be sent using special low-bandwidth out-of-band ICMP messages between routers. These messages between neighbors can be authenticated with the use of a TTL of 255, as in [3] and are timed out periodically (every minute) unless refreshed.

Preliminary Results. We provide a brief set of results regarding local profile construction and attack detection functionality described above. We use traffic generated by popular attack tools like `TFN` and `Trin00` along with traffic traces from Abilene backbone routers in NS-2, varying the number of spoofed bytes in source IPs and the attack rate against destinations of different levels of popularity. Our results show that the profiles generated by our sampling schemes are very stable and accurate across time over one hour periods. The fingerprinting schemes also have a very low false positive rate (we use $z = 1$) of about 2% for unpopular destinations and about 6% for popular destinations. In addition, for unpopular destinations, irrespective of the number of spoofed octets or the rate ("reasonably" high) of attack traffic, the false negative rate is close to zero. For popular destinations, the false negative rate is about 20% for low-rate, yet significant, attacks but improves rapidly as the rate of the attack is increased (uniformly true) across varying number of spoofed source IP octets. These results for the fingerprinting algorithms are very encouraging. We are yet to experimentally analyze the consensus algorithm.

3 Discussion

We believe that the above schemes can detect subtle and irregular changes in traffic patterns which may not be obvious from volume alone. For example, the subnet count fingerprints provide valuable information to distinguish a flash crowd from a DDoS attack [5].

It is hard for an attacker aiming to orchestrate a DDoS attack on a single machine or on the ISP infrastructure to circumvent the above mechanism. Since the fingerprints at each router keep track of details like the flow-length distribution and the traffic due to various source subnets, an attacker that uses arbitrarily spoofed source IP addresses on his attack packets makes detection/trace-back even simpler. Even if the attacker only spoofs a small number of source IPs and still orchestrates a heavy attack, the attack will be detected at routers where the attack traffic significantly impacts the normal operation of the ISP network, albeit with low confidence. Moreover, the consensus algorithm helps improve the detection accuracy.

Notice that it is possible for an attacker to "train" the normal traffic profiles over time to identify attack traffic as legitimate. We are investigating the balance between reacting to persistent changes in traffic and being susceptible to this form of attack. Also, the randomization of the measurement intervals (with a mean of one minute) helps ensure that the attacker cannot escape detection by spreading attack traffic over consecutive intervals to avoid detection in either interval.

We are investigating several important questions that still need to be addressed. These include identifying attacks that could avoid the measure profiles, measuring the exact space/computation requirements on modern router architectures and measuring conver-

[2] This information is obtained from the base profile.

gence/effectiveness of the consensus algorithm. We are also considering partial or incremental deployment issues such as identifying which subset of routers provide the best detection and the impact of partial deployment on the consensus algorithms. Finally, we plan to validate these algorithms by running them on real attack datasets.

References

[1] ALON, N., MATIAS, Y., AND SZEGEDY, M. The space complexity of approximating the frequency moments. In *Proceedings of STOC* (1996), pp. 20–29.

[2] ESTAN, C., AND VARGHESE, G. New directions in traffic measurement and accounting. In *Proc. SIGCOMM 2002*.

[3] FLOYD, S., BELLOVIN, S., IOANNIDIS, J., KOMPELLA, K., MAHAJAN, R., AND PAXSON, V. Pushback messages for controlling aggregates in the network. *draft-floyd-pushback-messages-00.txt* (2001).

[4] GIBBONS, P., AND TIRTHAPURA, S. Estimating simple functions on unions of data streams. In *Proc. SPAA 2001*.

[5] JUNG, J., KRISHNAMURTHY, B., AND RABINOVICH, M. Flash crowds and denial of service attacks: Characterization and implications for cdns and web sites. In *Proceedings of the 11th WWW Conference* (Honolulu, HI, May 2002).