Using Selective, Short-Term Memory to Improve Resilience Against DDoS Exhaustion Attacks

Qi Liao, David A. Cieslak, Aaron D. Striegel, Nitesh V. Chawla
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556 USA
Email:{*qliao, dcieslak, striegel, nchawla*}@nd.edu

Abstract

Distributed Denial of Service (DDoS) attacks originating from botnets can quickly bring normally effective web services to a screeching halt. This paper presents SESRAA (SElective Short-term Randomized Acceptance Algorithms), an adaptive scheme for maintaining web service despite the presence of multifaceted attacks in a noisy environment. In contrast to existing solutions that rely upon "clean" training data, we presume that a live web service environment makes finding such training data difficult if not impossible. SESRAA functions much like a battlefield surgeon's triage: focusing on quickly and efficiently salvaging good connections with the realization that the chaotic nature of the live environment implicitly limits the accuracy of such detections. SESRAA employs an adaptive *k*-means clustering approach using short-term extraction and limited centroid evolution to defend the legitimate connections in a mixed attack environment. We present the SESRAA approach and evaluate its performance through experimental studies in a diverse attack environment. The results show significant improvements against a wide variety of DDoS configurations and input traffic patterns.

I. Introduction

With the advent of tools and scripts that trivialize the creation and management of large numbers of compromised hosts (aka botnets), low profile armies exist that are ready to launch devastating DDoS attacks at any moment. The sheer prevalence and highly distributed nature of the hosts further complicates matters. Hence, considerable research attention has been brought to bear on how to effectively prevent, detect, defend, and/or neutralize the botnet threat [1]. Moreover, the diversity of botnets in the 'wild' necessitate multi-tiered strategies in order to effectively weather attacks.

In this paper, we specifically examine the issue of end host defense against such a distributed resource exhaustion attacks with non-spoofed legitimate requests. Based on the vast amount of data that the servers observe daily, can one use lightweight data mining techniques to enable mitigation mechanisms that reduce or nullify the offending botnet(s)? We assume that the end service operates from the perspective of isolation with no external information services beyond those directly initiated by itself are available. As the issue of end host defense has roots in load balancing and anomaly detection, we note the key novelties of our work versus that of the large body of existing work [2]–[11]:

- *Effectiveness of short-term data views:* We note that short-term pattern extraction (extracting patterns in the order of minutes versus hours or days) can be effective against botnets. Rather than attempting to extract the long-term 'clean' signal, short-term extraction meshes well with open sites whose input traffic is highly variant (for example eBay, CNN, etc.).
- *Pattern evolution as an input:* While short-term pattern extraction provides an excellent baseline, the evolution of the extracted patterns (for example clusters) is interesting. We introduce the notion of adaptively skipping between past patterns under sustained attacks to prevent attacker profiling and comment on using evolution as a secondary filtering input.
- *Quality metric - Session Completion Time:* While intuitively understood, we note the distinctive performance effects of considering the satisfaction of multiple TCP flows (for example web traffic) as requisite for a successful session (group of flows). We introduce the notion of Session Completion

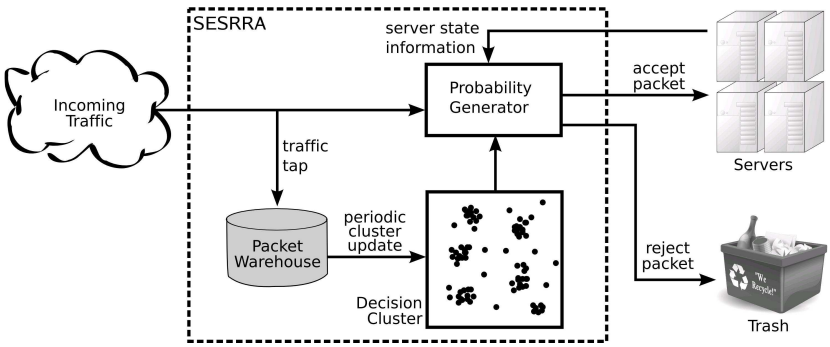

Fig. 1. View of SESRAA co-located with a load balancer.

Time (SCT) for evaluating botnet performance mitigation. To the best of our knowledge, ours is the first study to consider session-wise rather than connection-wise QoS.

We describe SESRAA (SElective Short-term Randomized Acceptance Algorithms), a host-based scheme that employs a nimble and adaptive response mechanism to mitigate malicious attacks in a noisy, live environment. Our solution is co-located with the traditional load balancer (see Figure 1). Rather than attempting to remove the noise inherently present in the traffic, our scheme embraces the noise and adapt rapidly to network dynamics. The design of our solution is centered around a difficult problem faced by research community: *How should a system react when attacks are massive, distributed, rapid, dynamic, and there exists little to no pattern of what constitutes good behavior?* To that end, SESRAA attempts to offer an effective first line of defense to handle short-term mitigation while more sophisticated but slower schemes (pushback, tracing, etc.) run their course. Through the adaptive application of k-means clustering and recording of the cluster evolution, SESRAA consistently changes its defense profile to both better isolate ongoing attacks and to reduce its vulnerability to profiling attacks.

The remainder of the paper is organized as follows. Section II differentiates our research approach with existing work and acknowledge the previous studies in the field. In Section III we discusses the environment in which our solution operates and theory behind our approach. Detail explanation is given to how the clustering should work and how the suggested probability should be generated. Section IV shows some traffic analysis and the evaluation of proposed solution to DDoS attacks via simulation. Further experiments is studied and discussed in Section V. Finally, Section VI offers several concluding remarks and comments on the ongoing future work for SESRAA.

II. Related Work

Traditionally, Distributed Denial of Service (DDoS) attacks have been characterized as either TCP SYN or UDP data flooding. In the case of TCP SYN flooding, the three-way handshake is never completed (SYN flood) or the handshake completed but data never sent. In contrast, UDP data flooding simply overwhelms the link in question through sheer bandwidth with the source address often spoofed to hide the location of the source(s). Critically, a first step in defense is the identification that an attack is occurring versus the server simply being overloaded due to popular content. In [19], the concept of a one-way connection density (OWCD) function is used to describe the asymmetric property of incoming and outgoing traffic for detection of DDoS attacks. Similarly, the basic assumption in [20] is that the ratio of incoming traffic over outgoing traffic is high during the DDoS period because the server could not respond quickly under heavy load. Kang, Zhang, and Ju proposed the use of hierarchical clustering in [21] to classify the type of DDoS attacks. Conversely, our work focuses on the initial mitigation mechanisms rather than detection but does not preclude the use of existing work with regards to DDoS attack detection as a trigger mechanism rather than solely using system load.

As flooding typically involves spoofing the source of the packet, numerous works have attempted to reduce or eliminate spoofed packets. Xu and Lee propose in [22] is to filter spoofed packets through URL redirection via MAC verification but require router support to provide such a mechanism. In [23], Kim et. al propose a packet scoring mechanism whereby packets are discarded based on a per-packet score threshold. Yaar, Lessig, and Song propose SIFF in [24] which controls Internet flows based on the notion of privileged or unprivileged channels. In a similar vein, Path Identification [6] is a packet marking technique that embeds a path fingerprint within each packet by altering the IP Identification field. This allows a victim to identify packets traversing the same paths through the Internet on a per packet basis, despite spoofing. Path Identification can remain effective in a partial deployment, up to 50%, but still suffers from the cost of deployment on a large scale.

In the case where attacks are not spoofed, an approach to curtailing the DDoS attack problem is through IP Traceback [2]–[4]. The goal of traceback is to identify systems responsible for generating attack traffic and to determine the network path through which the malicious communication flows. However, in the botnet setting, attacks may involve involves hundreds if not thousands of machines intermixed with legitimate traffic making tracing back to individual hosts impractical. Park and Lee [2] present a discussion on the effectiveness of source identification through probabilistic packet marking. This technique may potentially reduce the DoS problem to a manageable level, however, this scheme suffers as spoofing of the packet marking field eliminates the ability of the method to perform an accurate traceback.

In addition to genuineness of source, DDoS can also be categorized into bandwidth depletion and resource depletion attacks. Bandwidth depletion attack can be mitigated by filtering unwanted traffic earlier which would have been discarded anyway later at the end server (for example UDP data flood). In [17], Mahajan et al. propose a pushback mechanism to control aggregates at the upstream router. The rate limiting scheme from the upstream gateway can be combined with our approach to provide a comprehensive end host defense against DDoS attacks. In a similar vein, level-k max-min proposed in [18] offers an alternative over the earlier recursive pushback mechanism when attacking hosts are spread across the network. However, the rate limiting requires the identification of the offending nodes in order to effectively contain the malicious traffic. We note that SESRAA could be applied as the input for rate limiting to blend the two techniques which could be a subject for future research.

A significant hindrance to many of these techniques discussed above is the requirement of extensive deployment throughout the router infrastructure of the Internet versus more deployable but typically less effective host-based solutions. The approach by Xu and Lee in [22] uses game theory to model the attackers and defenders and apply a quota to each client to confine the clients to their fair bandwidth share. In contrast, our work focuses on probabilistically filtering the likely botnet patterns rather than diluting the share of each client which we believe provides a more responsive solution. Specifically, our work incurs only a fixed state when in the overloaded state and only imposes increased state / load when the system is lightly loaded as opposed to potentially per-client state in [22]. Another promising victim-based DDoS attack detection scheme was proposed by Jin and Yeung in [25] using covariance models to combat SYN flooding attacks but unfortunately the work provided minimal mapping for how to apply the results with regards to mitigation. Jin, Wang, and Shin proposed an alternative filtering scheme based on hop-count filtering in [26]. This technique is able to allow a server to infer authenticity based on an analysis of expected and actual TTL values of packets from a single host. However, a study of the effectiveness of this technique against a DDoS featuring unspoofed traffic has yet to be performed.

The first large-scale study of the application of the subspace method to traffic flows was performed in [27]. Incorporating anomaly detection with this technique has yielded strong results, allowing analysts to note distinct traffic behavior such as flash crowds, worms, and even changes of routing policy. While this technique is not an automated solution, it demonstrates the power the subspace method in terms of network traffic identification. In Static Clustering (SC) and Network-aware clusters (NAC) [28], [29] clients are clustered by some common properties like the longest matched prefix/netmask BGP routing table snapshots. Finally, we also acknowledge another attack model called Reduction of Quality (RoQ) [30], which targets adaptation mechanisms and keeps an adaptive mechanism oscillating between over-

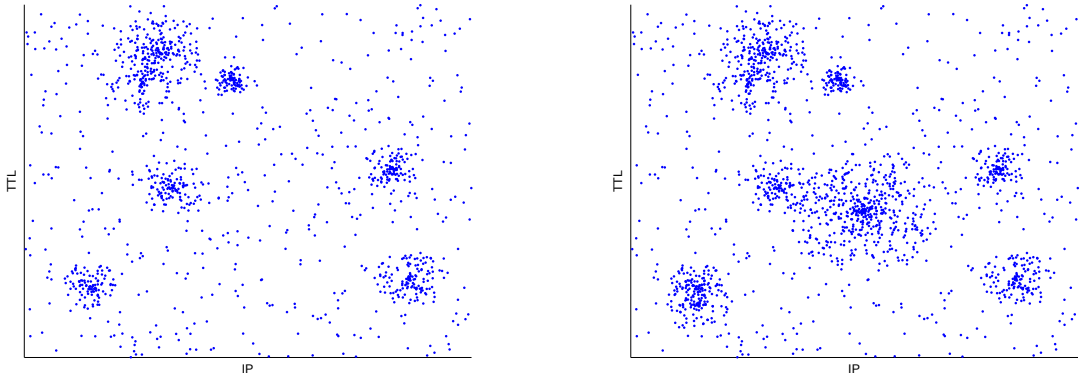

Fig. 2. Side-by-side analysis of connections characteristics. The left is a partial view of collection of connections immediately prior to the DDoS and the right is partway through the attack.

load and under-load conditions. Unlike DDoS, RoQ optimizes the attack traffic to produce the maximum damage while keeping a low profile to avoid detection.

III. Architecture Design of SESRAA

In the literature, DoS attacks typically employ IP and port spoofing to quickly overwhelm the server. Rather than addressing the removal of spoofed IP addresses, we assume that the handshake portion of the TCP connection can be largely offloaded to the load balancer, thus only creating load in the bandwidth rather than computational sense. In contrast, we focus on a more difficult problem in which a series of hosts attempt to overwhelm the server through legitimate connection requests (i.e. handshake is completed and all data is downloaded). This *resource exhaustion* attack is made possible with the recent emergence of large scale botnets in which vast amount zombies do not even bother to hide their real addresses. The threat model we are considering is that an attacker, such as a botnet master, controls tens of thousands of compromised machines and makes *legitimate* connection requests to victim machines, such as a web server, thus making its resources unavailable to valid users. Hence, SESRAA targets the mitigation of botnet-style DDoS exhaustion attacks wherein the attack consists of a burst of activity to the target victim.

Our conjecture is that despite the noise of the Internet, there is good behavior to be extracted over short periods of time. To that end, we consider *k*-means clustering [12] to discover clusters of normal activity or behavior based on connection requests. This premise further bolstered by observations in [13], [14], which noted that connection requests tend to cluster over time. Intuitively, if the normal traffic tends to self-organize in cohesive clusters, the overload state will introduce anomalous or malicious behavior that falls outside of clusters or overloads existing clusters. Conversely, traffic that falls within a cluster has a higher probability of being good, until the cluster density increases beyond an acceptable load. By probabilistically filtering attackers based on the weighting of the clusters and a distance function, SESRAA infers the potential for a connection to be bad in a lightweight and efficient manner. Moreover, clusters are continually recomputed with multiple characteristics, which presents an attacker with a non-static defense scheme.

A. Operation of SESRAA

When the server (S) reaches a warning state (S_{warn}), the most recent set of centroids M derived from *k*-means clustering over IP and TCP characteristics freezes. The proportion of distributions around some members of M will rise as other fall while connection requests flood the system, which is seen in Figure 2. New connection requests are placed into clusters, which will cause a disruption to the population balance, allowing SESRAA to probabilistically adapt to the attack in order to better serve members from

Server Load	Server Action	SESRAA Action
Overload (S_{over})	Maximum probabilistic connection rejection	Trigger adaptation Clustering
Warning (S_{warn})	Probabilistic rejection of new connections	Clusters Fixed
Normal (S_{normal})	All connections accepted	Recompute clusters at intervals

TABLE I

SESRAA AND SERVER ACTIONS WITHIN EACH STATE

the good clients (C_G). Based on the load to the server (S), the differential between the actual and expected cluster size, and the relative distance (based on deviation and cluster density) from the cluster's centroid, an acceptance probability is assigned. This method allows the isolation of collections of flows that are different from those found when the traffic was in a normal state. Ideally, such flows are those of the bad clients (C_B). As the botnet attacks recedes, the server reverts to a normal state (S_{normal}), allowing it to prepare for the next attack. This offers a unique, adaptable, and resilient method to managing a botnet attack while still delivering service to legitimate users. Table I summarizes the actions taken at SESRAA during different sever loads.

While the normal state (S_{normal}) does not imply that all accesses are good, we believe that one can infer that a higher probability of goodness for connections received during this state. However, SESRAA can be tuned to not use this metric as the sole test for goodness as this assumption possesses two key weaknesses. First, an attacker may be conducting a low level botnet attack that does not overwhelm the server. Second, a cluster of hosts that was previously considered good may become compromised. In both cases, the simple metric of training the goodness of connections based the underload state may be deceptive. However, it is important to note that the increase in traffic that occurs with a botnet attack will likely cause previously noted *good* cluster(s) to become overweighted and thus subject to increased rejection probabilities. Even a slow ramp-up for the botnet attack is not immune to rejection by overweighting as the cluster weight for the botnet cluster will still manifest itself by the magnitude of the number of connections in the cluster.

To further defend against those bots that attempt to poison the clustering, $W_{cluster}$ and $W_{centroids}$ are used for the evolution of cluster and centroids in SESRAA. For each window size of $W_{cluster}$, the cluster's centroids are recomputed and all traffic data points are discarded. These centroids will be used as the fresh starting points in each cluster for the new round of $W_{cluster}$. $W_{centroids}$ is used to keep a history of recent past sets of computed centroids. In the event that the current clustering techniques do not mitigate the overload state (S_{over}) at the server due to the insufficient learned pattern, a random hop between these centroid sets at fixed intervals is adopted until all traffic move to S_{normal}. Overall, these adaptation mechanisms provide a non-deterministic rejection method to botnet. Our future work will explore a method of differential of clustering for botnet attacks.

B. Rejection Probability Generation

When the data is manageable, these requests are merely noted as examples with which clusters are established. However, when the server is in danger of failing, SESRAA needs the ability to distinguish traffic falling within the accepted pattern. Thus, SESRAA's task becomes that of outlier detection. To perform outlier detection, we were initially inclined to apply a method that incorporates Local Outlier Factor [15], which is a useful measure of both a point's distance from a cluster and the relative measure of the cluster's density. However, such an operation is computationally infeasible given the requirement to calculate Nearest-Neighbor (NN) for each connection and the small quantum of time budgeted to accept or decline the connection. Thus, our system must rely on a faster heuristic for assigning a probability that the connection belongs to a given cluster.

As the exact distribution of connections is unknown, we rely upon Chebyshev's theorem stating that no more than k^{-2} of the examples exists at a distance k standard deviations or greater from the mean

[16]. This is invaluable in determining acceptance probabilities based on the cluster information we have gathered, as this model makes no assumptions about the distribution of connections within each cluster. For each new connection received when the server is under attack, Chebyshev's theorem suggests:

$$P_{cluster}(X) = \frac{\sigma^2}{(\mu - X)^2}$$

where μ represents the cluster mean, X represents the connection datapoint, and σ represents the cluster standard deviation. As new traffic enters the connection subspace, those points distanced furthest from cluster centers will receive the lowest acceptance probability (such as the sparse points distant from centroids), while those closer to cluster centroids will be increasingly accepted.

As the load on the server increases, the policy also needs to become much harsher to prevent server overloading. The primary danger of the botnet attack stems from overloading the capacity of the server. Given two scenarios where the server was presented an identical connection request and the distributions and partitions were identical between scenarios, the Server cannot issue the same acceptance probability if one scenario features a differing traffic load than the other. As the server reaches its resource capacity, the rate of rejection must rise significantly. This implies that any rejection probability component derived from load over threshold will follow an exponential growth pattern. Therefore, the acceptance probability function based on load is:

$$P_{load} = e^{-\left(\lambda \cdot \frac{l(t) - l_{thres}}{l_{thres}}\right)}$$

where λ is a constant weight, $l(t)$ is the load on the server at time t and l_{thres} represents the server's threshold load.

The final considerations of the acceptance probability generation is where a connection maintains the existing distribution of connections around each cluster. If a connection causes a cluster to deviate over its natural distribution against the entire population, then the probability of its rejection should be increased. For distribution consideration, it is sufficient to calculate a probability based on the separation from the actual and expected distribution of the cluster.

$$P_{dist}(X) = \frac{d_{exp}}{d(X)}$$

where $d(X)$ represents the percent of all packets within the cluster of connection packet X and d_{exp} represents the percentage of connections within the given cluster at the point when the clusters are fixed. This component of probability generation will therefore assign connections falling within heavy regions to very low acceptance probabilities as these localized changes to the overall traffic aggregate likely reflect a botnet attack.

While it is straightforward to merge the above functions to calculate an acceptance probability during S_{over} using a linear combination of weights and probabilities, this metric has some shortcomings. First, after an extended period of attack, ideally the weighting can adapt to focus on $P_{cluster}$, P_{dist} or P_{load}. However, the dynamically adjusting the weights entails extra computation cost. Second, the magnitude of rejection does not increase significantly using the linear relation, which is particularly important within the context of a botnet attack. Therefore, for simplicity and effectiveness, product of the above probabilities is adopted to calculate the final probability, which the load balancer then uses to decide connection acceptance or rejection. Although a discrete accept/reject model is nearly infeasible, such a probabilistic heuristic should prove to be an effective method to mitigating a botnet attack.

C. Discussions

Beyond the core operations of SESRAA, we discuss the following items in more detail: flash crowds, false positives, and the computation cost of SESRAA.

Flash Crowds: While one can debate whether *flash crowds* can be defined as "good" or "bad" behaviors, the reality is that flash crowds are generally not intentionally malicious, rather the requests are driven by exceptionally popular content. Although SESRAA's discerning offers limited benefit in discerning the participants of the flash crowd versus the normal traffic behavior, SESRAA will still control admissions to ensure that those connections admitted can indeed be served in a timely manner. SESRAA does have the capacity to limit highly localized flash crowds (for example university message board posting) which can be a desirable outcome to encourage caching on the part of the remote network. In particular, we note that SESRAA will not do any harm beyond what a normal load balancer would employ if all requests are considered 'good' in the larger network context.

False Decision: For a system without perfect knowledge of request intention, false positives and false negatives are a practical reality of the system. Critically, SESRAA is targeted at systems where the noise of the Internet precludes the creation of extremely accurate models, i.e. the users of a site change over time rather than exhibiting an easily extractable behavior such as with eBay, CNN, Amazon, and university web sites rather than an authorization-driven web site such as scientific portals and others. We accept that SESRAA will clearly make incorrect decisions but that tradeoff is taken in exchange for nimbleness of the system (preventing simply profiling attacks via adaptation) and elimination of long-term training periods (simply unfeasible). As noted in our upcoming experimental studies, SESRAA offers significantly improved results over the merely randomized load balancing demonstrating a clear improvement in false positives / false negatives. Hence, we believe the benefit gain in our approach outweighs the incorrect decisions made by the server in accepting/rejecting clients requests during the DDoS period in order to maintain a reasonable load with the realization of good QoS for the majority of legitimate users.

Computation Cost: It is important to note that the core of SESRAA are relatively lightweight data mining operations that are conducted at the load balancer, not the web server itself. Moreover, the transition to a highly loaded state freezes the set of centroids (recent and past) used for evaluation and does not incur a data mining load when the system itself is under attack. These computations could also be directed to a separate server besides the load balancer if the load balancer does not have sufficient resources when under a light load. While we do acknowledge that the k-means clustering process is not deterministic, the lack of determinism is mitigated by its invocation during only periods of low load. The computation during the rejection phase is also relatively lightweight as the Euclidean distance versus existing clusters is a constant factor that does not require significant overhead and creates a fixed impact per connection.

Bandwidth Exhaustion Attack: Finally, we note that SESRAA does not mitigate pure bandwidth exhaustion attacks (for example UDP flood). SESRAA solely focuses on the issue of botnet-style resource exhaustion via legitimate requests for which traditional SYN-flood pushback-style mechanisms [17], [18] would offer little benefit. SESRAA does not preclude the use of these techniques to filter out unwanted traffic and is largely complementary to those techniques.

IV. Simulation Analysis

To understand the traffic pattern and to evaluate SESRAA, we performed simulation and experimental analyses that are discussed in the next two sections. To start, we conducted a simulation analysis based on a month of live captured data from the primary web server on the university campus. The usage of actual data allows for the avoidance of artifacts of random addresses, random ports, and other traffic characteristics that would be trivial to extract via data mining. Moreover, the usage of actual data allows for the accurate simulation of members of a 'live' botnet with real addresses, ports, TTLs, and request characteristics that represent subverted hosts from the observed traffic flows. While the usage of actual data rather than synthetically data increases the difficulty of defense, we firmly believe that our results more accurately reflect what would occur under an actual botnet exhaustion attack. The university tap data was used both for the simulation and experimental analyses.

To enable testing across a wide variety of scenarios, a discrete-event simulator for SESRAA was

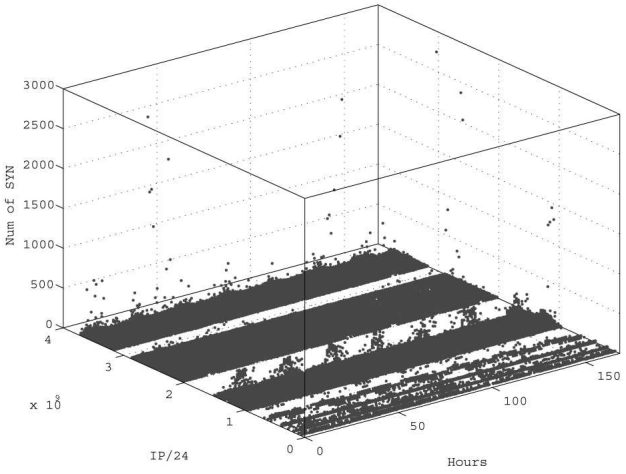

Fig. 3. A 3-D view of the number of connections of all incoming web traffic for a seven days monitoring period. X axis is time in unit of hours; Y axis is the IP with the first three significant bytes; and Z axis is the total number of connection requests during that hour.

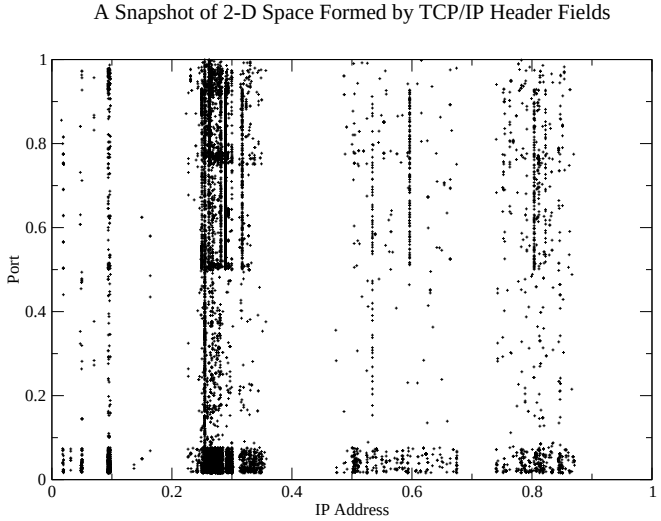

Fig. 4. A 2-D view of incoming web request for a monitoring period of approximately one hour. X axis is the source IPv4 addresses normalized to 2^{32} and Y axis is the source ports normalized to 2^{16}.

developed in Java[1]. The clustering criteria uses characteristics from the packet headers of both the network and transport layer, which includes but not limited to IP address, port numbers, TTL values, and sequence numbers. While Euclidean distance of multi-dimension is easy to compute and is actually adopted by our approach, for visualization easiness, we present results using only the *src_ip* and *src_port* to form a 2-D space for *k*-means clustering.

With regards to clustering, a sliding window of 55 simulation seconds is used. If there has not been a recent attack, the data points of the current clusters will be dropped and only the centroids themselves will be saved. In the simulation, the cluster centroids are updated each simulation second provided that the server is under the appropriate load level. Client-side characteristics are modeled with a 100 Mb/s link and varying WAN-side RTTs. The load and SESRAA threshold are set 0.8 and 0.6 respectively, denoting the load at which the respective algorithms are invoked. The timeout at the server is set three seconds while the client SYN timeout set to be two seconds with a maximum of three retries before declare failure. The policies of load only (probabilistic rejection beyond a threshold), SESRAA (outlined earlier), and perfect (always reject attackers) are compared.

The primary performance metric, quality preservation in terms of Session Completion Time (SCT), is defined in the following manner:

$$SCT = Max(FCT_{f_0}, FCT_{f_1}, ..., FCT_{f_n})$$

where FCT_{f_n} is the flow completion time of flow n. In order for a session to be successful, each flow within the session must complete its gathering of data before the timeliness requirement of the client. For instance, a real web session might require three connections to render a webpage, and each of which must complete in their entirety. A session is then determined to be *successful* if the $SCT \leq T_{QoS}$, where T_{QoS} is the QoS constraint of the user. Session-wise QoS is more important than connection-wise QoS because good connection-wise QoS does not necessarily guarantee that session-wise QoS will be satisfied. In the case of load balancing, this is especially critical as a rejected connection will prevent the completion of a successful session. From the session completion metric, the percentage of good hosts serviced successfully serves as an effective metric to evaluate the effectiveness of botnet mitigation. If not enough 'bad' users

[1]Available at netscale.cse.nd.edu

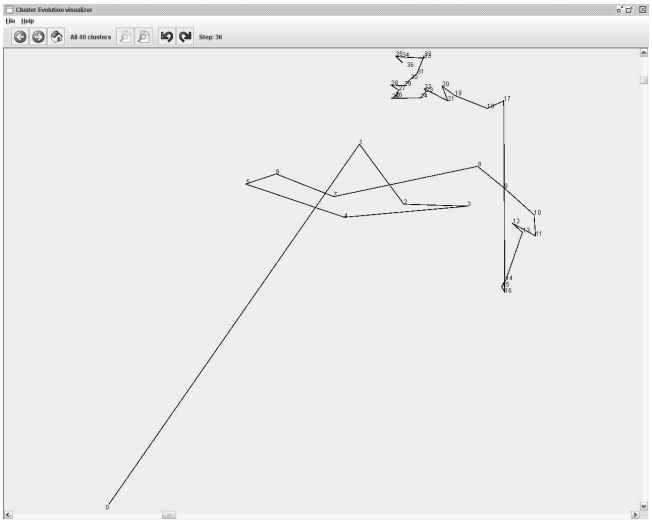

Fig. 5. Screenshot of clusters evolution visualizer (zoomed-in for the centroids movement within one chosen cluster)

are prevented from accessing the server, the QoS for all sessions will suffer. Consequently, if a load balancing scheme is indiscriminate, individual connections of good users may be rejected which prevents the completion of the overall session.

To that end, we studied the hourly, daily, and weekly traffic pattern of all incoming web requests to the main web sever on campus. Figure 3 shows a seven-day traffic pattern of all incoming web requests to the main web sever on campus. While the majority of total number of connections for a specific subnet (/24 is the default unit used) within an hour is below ten, there are some extremes observed in certain /24 IP addresses. A diurnal pattern can also be observed for some IP ranges which is an interesting observation that is not currently accounted for in SESRAA but is an open topic for future research.

A scatter plot of the 2-D space used for clustering formed by the source IP addresses and port numbers is shown in Figure 4. It provides a snapshot of about one hour of web traffic coming to the web servers on campus. Short-term clustering on IP and port shows that port numbers are heavily distributed at low and high ends for certain IP range. A simulation visualizer (Figure 5) was developed to visualize the clusters and centroids evolution. One can zoom in each individual cluster and see the centroid movement within that cluster by stepping through each time unit, or we can get an overview of the distance between different centroids of clusters. The visualization helps one to understand the potential pattern and provide better adaptation to botnet attack behavior.

A comparison of QoS performance of various policies are summarized in Figure 6. The Perfect policy is the oracle that knows exactly which SYN packets are good or bad, i.e. the equivalent of an 'evil' bit in the packet header. The Load policy represents a nominal load balancer that exemplifies a traditional load balancer. We note that in the figure, the Load policy performs relatively well at the initial phase of the botnet attack as the existing connections are still legitimate hosts. Once the legitimate hosts finish their existing connections, the overwhelming nature of the botnet drastically reduces the rate of good connections having their session-wise QoS satisfied despite having the same relative request rate as before the initial attack. In contrast, SESRAA offers a rate of successful sessions for good clients roughly halfway between the Perfect and Load policies with a k setting of $k = 10$. Clearly, SESRAA is incorrectly rejecting a portion of good connection requests but serves its purported role, i.e. to act as a triage, salvaging part but not all of the connections to in essence buy time for other mechanisms to function. By the end of the DDoS period, SESRAA has provided a roughly 25% cumulative improvement over the normal load balancing scheme in terms of good users with their sessions successfully satisfied.

While setting $k = 10$ offers reasonable performance, an interesting question is posed with the ideal setting of k.

In Figure 7, possible choices of *k* values are studied with the same data input as with Figure 6. Normally,

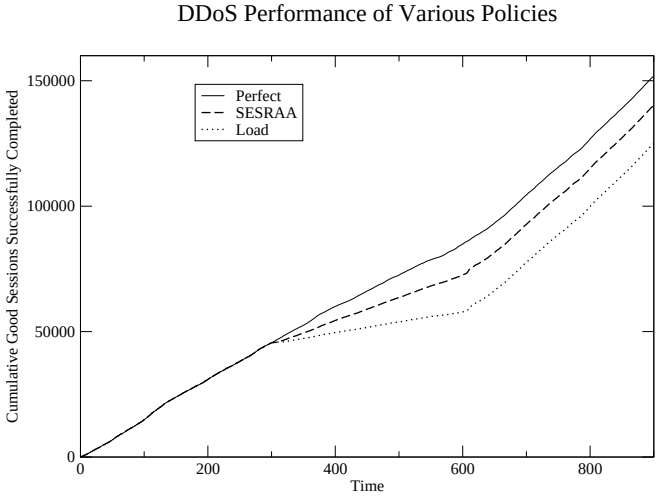

Fig. 6. Comparison of QoS Performance of Perfect, SESRAA, and Load Policy Only in a Random Botnet attack (time 300 to time 600).

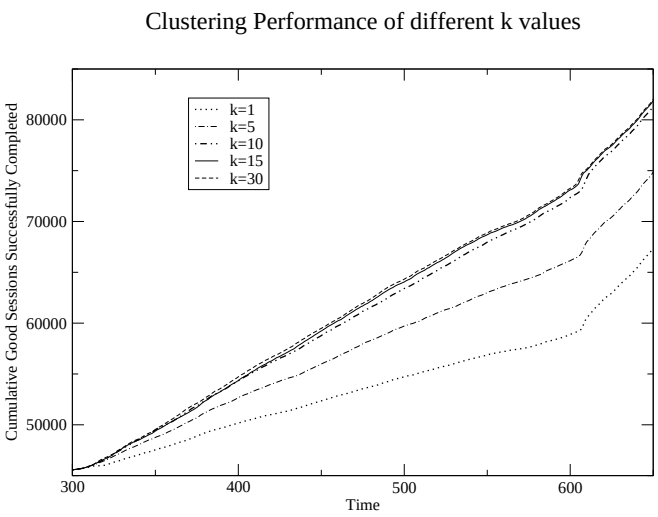

Fig. 7. Performance of Choices for Various *K* values during the attacking period (time 300 to 600).

k should be increased with higher connection rate depending on the server's traffic volume. One possible choice of *k* is to minimize the cost function $C(distortion + MDL)$. The $distortion$ function is defined as $\sum_{j=1}^{k}\sum_{i=1}^{|C_j|}(x_i - c_j)^2$, where C_j is the *j*th cluster, x_i is a data point in each cluster and c_j is the centroid in the cluster. Minimum description language (MDL) is defined as $\lambda(m \cdot k \cdot logn)$, where λ is a predefined constant, *m* is the number of features (IP, Port, TTL, etc), *k* is number of clusters and *n* is number of instances in training data set. The choice of λ depends on each web application. To avoid overfitting, we need to reduce the number features and the number of clusters. As we can see from Figure 7, one cluster (*k*=1) has the worst performance as the single cluster is clearly insufficient to fit the data. Conversely, a larger k improves performance but only up to a certain level as shown by how how a k value of 30 is only slightly better than the initial k value of 10.

V. Experimental Studies

In addition to our traffic analysis and simulation studies, a preliminary version of SESRAA was developed to validate our initial results. In our experiments, the environment is divided into four components: clients, emulated network, load balancer, and the server. The clients are responsible for generating connection requests via *wget* to the server. In the emulated network, connection requests are mapped to a characteristics space derived from live university traffic from network tap. The list of potential clients was separated into groups of good and bad clients with the bad clients elected to conduct a botnet attack during the experimental period.

The experimental testbed was connected by a 100 Mb/s switched network with Layer 2 overlay connectivity provided by *libpcap*. The web content served reflected a CNN style page: a 66 KB hypertext page featuring several small pictures. All content retrievals were conducted using *wget* which employed HTTP 1.0 (separate connections for each object). The DDoS attack lasted 150 seconds for all experiments. We selected HTTP 1.0 as it is significantly more difficult than HTTP 1.1 due to the fact that the denial of any single connection for a good client (i.e. image 1 in the webpage) will cause the good client session to be counted as failed.

We define different scenarios of attack for each experiment separately. In the simplest generation method (Random DDoS), attackers are selected randomly from the client space; in essence, a random set of clients from C form C_b. A more sophisticated attack (Clustered DDoS) represents a localization within the connection space. Here, a single centroid is picked within the connection space. Members from C are more likely to join C_b the closer they are to the selected centroid, generating a singular attack cluster. The

most sophisticated form of attack (Homologous Clustered DDoS) mimics the pattern of legitimate traffic, making differentiation between good and bad clients nearly impossible. While such an attack is extremely unlikely as the attacker would need to have inside information (i.e. access to the tap), we present this scenario to demonstrate that SESRAA can still offer a benefit even in this worst case.

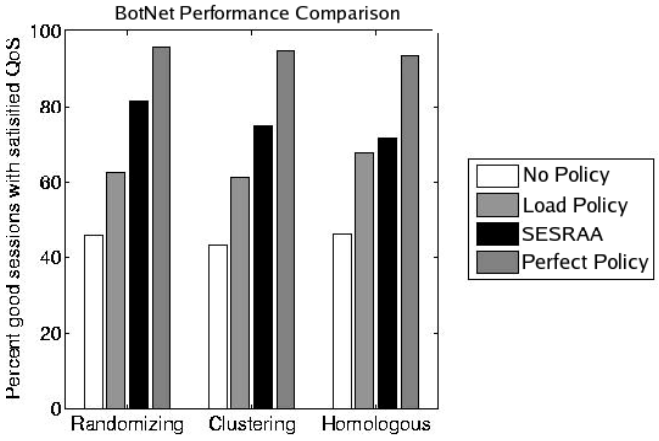

Fig. 8. Summary of QoS preservation (100% represents all legitimate hosts serviced)

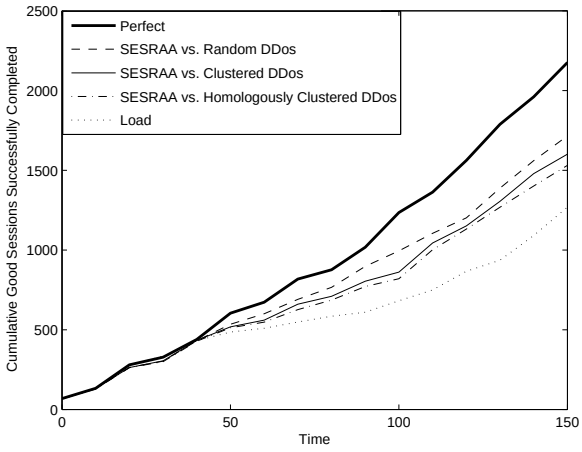

Fig. 9. Cumulative total of good sessions successfully completed

Figure 8 indicates that as expected, the overall Quality of Service based on our metric falls between the load policy and perfect policy. The figure itself shows the end result of performance after the test has completed. In the first case of randomized attackers, SESRAA performs best, noting the discrepancies as the clients in the botnet themselves tend to fall into the outlier range and have a higher rate of rejection. The introduction of clustering to the clients reduces the performance of SESRAA slightly as the request is less likely to fall into the outlier category and must overload a cluster first before being rejected. The cluster type of attack does require some intelligence in the attack and a reasonable approximation or biasing of the clusters to group the attackers into a single cluster. In the third case, SESRAA still outperforms normal load balancing but only slightly due to the fact that the botnet traffic is nearly perfectly balanced among the normal clusters that would be derived. In practice, such an attack is extremely unlikely as it would require that the attacker have precise information about all connections (i.e. a tap to all network data to the server) in order to appropriately guide their botnet with regards to attacking.

The graph in Figure 9 represents several scenarios that a server may face. This case generates independent clusters of attack traffic. This represents the case in which a number of similar machines are compromised to form the zombie army. Suppose a Trojan horse compromised a university's residential computing network. As most machines are likely Windows machines, leading to similar TTL values, for personal use and the range of addresses is relatively tight, a botnet attack generated from such a collection would likely generate a separate cluster than that of the normal traffic. Figure 9 demonstrates the domination of the perfect policy curve over both the load and SESRAA policies. While SESRAA outperforms the load policy, the degree of domination is less than in the case of the purely randomized DDoS. Although SESRAA assigns high probabilities to connections falling within attack clusters with sufficient separation from legitimate traffic clusters, overlapping attack and legitimate traffic clusters lead to a reduced rate of attack connection rejections. In an applied setting, if there is a grouping of legitimate machines, such as a university cluster, for a SESRAA server's content and then a compromising within this group occurs, SESRAA will have difficulty in distinguishing between attack and legitimate hosts within this cluster. On the basis of its bias, SESRAA's performance will suffer, particularly in instances of overlap. In such a case, we believe the use of cluster evolution in a combined heuristic with the basic SESRAA approach could provide interesting insight. The inclusion of evolution directly into the heuristic is an on-going work.

VI. Conclusion and Future Work

In this paper, we present SESRAA, an effective probabilistic connection rejection method that provides management against botnet-based DDoS attacks. Our technique employs k-means clustering while there is a manageable load on the server. As the load becomes unbearable, the server then applies several metrics based on distance from cluster center and cluster population and uses these to assign rejection probabilities accordingly. Using this method will assist in isolating connections likely to be attacks and will enhance the Quality of Service delivered to legitimate clients. Once the attack has passed, the server reverts to its learning state.

The results of our experiments indicate that this is an effective technique, which leads to several other avenues of study. We intend to perform a full-scale study of temporal network traffic and determine methods of segmentation, using both clustering and hyper rectangles, and visualization, which will be useful in better understanding network traffic, particularly during a botnet attack. Further works includes expanding the usage of cluster evolution as well as long-term studies of network traffic.

Acknowledgements

The authors would like to thank the anonymous reviewers for their insight and comments on this paper.

References

[1] T. Holz, "A short visit to the bot zoo," *IEEE Security & Privacy*, vol. 3, no. 3, pp. 76–79, 2005.

[2] K. Park and H. Lee, "On the Effectiveness of Probabilistic Packet Marking for IP Traceback under Denial of Service Attack," in *Proc. of INFOCOM 2001*, 2001, pp. 338–347.

[3] A. Snoeren, C. Partridge, L. Sanchez, C. Jones, F. Tchakountio, S. Kent, and W. Strayer, "Hash-Based IP Traceback," in *Proc. of SIGCOMM*, 2001, pp. 3–14.

[4] S. Savage, D. Wetherall, A. P. Karlin, and T. Anderson, "Practical Network Support for (IP) Traceback," in *Proc. of SIGCOMM*, 2000, pp. 295–306.

[5] J. Ioannidis and S. M. Bellovin, "Implementing Pushback: Router-Based Defense Against DDoS Attacks," in *Proc. of Network and Distributed System Security Symposium, Catamaran Resort Hotel San Diego, California 6-8 February 2002*. The Internet Society, February 2002.

[6] A. Yaar, A. Perrig, and D. Song, "Pi: A path identification mechanism to defend against DDoS attacks," in *IEEE Symposium on Security and Privacy*, May 2003, pp. 93–107.

[7] A. Keromytis, V. Misra, and D. Rubenstein, "Sos: An architecture for mitigating DDoS attacks," *IEEE Journal on Selected Areas of Communications (JSAC)*, vol. 22, no. 1, pp. 176–188, Jan. 2004.

[8] T. Peng, C. Leckie, and K. Ramamohanarao, "Protection from distributed denial of service attack using history-based ip filtering," in *IEEE International Conference on Communication (ICC)*, vol. 1, May 2003, pp. 482–486.

[9] H. Wang, D. Zhang, and K. Shin, "Detecting SYN flooding attacks," in *Proc. of IEEE INFOCOM*, 2002.

[10] A. Hussain, J. Heidemann, and C. Papadopoulos, "A Framework for Classifying Denial of Service Attacks," in *Proc. of ACM SIGCOMM*, 2003, pp. 99–110.

[11] Z. Shu and P. Dasgupta, "Denying Denial-of-Service Attacks: A Router Based Solution," in *International Conference on Internet Computing 2003*, 2003, pp. 301–307.

[12] J. MacQueen, "Some methods for Classification and Analysis of Multivariate Observations," in *Proc. of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1965, pp. 281–297.

[13] P. Chhabra, A. John, and H. Saran, "PISA: Automatic Extraction of Traffic Signatures," in *Proc. of IFIP NETWORKING*, 2005, pp. 730–742.

[14] A. Lazarevic, L. Ertoz, V. Kumar, A.Ozgur, and J. Srivastava, "A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection," in *Proc. of 3rd SIAM Conference on Data Mining*, San Francisco, CA, January 2003.

[15] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," in *ACM SIGMOD Int. Conf. on Management of Data*, 2000, pp. 93–104.

[16] A. Mood, F. Graybill, and D. Boes, "Introduction to the Theory of Statistics 3d edition," p. 564, 1974.

[17] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxon, and S. Shenker, "Controlling high bandwidth aggregates in the network," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 3, pp. 62–73, July 2002.

[18] D. K. Y. Yau, J. C. S. Lui, F. Liang, and Y. Yam, "Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles," *IEEE/ACM Transactions on Networking*, vol. 13, no. 1, pp. 29–42, 2005.

[19] T. Xu, D. K. He, and Y. Zheng, "Detecting DDOS attack based on one-way connection density," in *10th IEEE Singapore International Conference on Communication Systems (ICCS)*, Oct. 2006, pp. 1–5.

[20] W. Lu and I. Traore, "An unsupervised approach for detecting DDoS attacks based on traffic-based metrics," in *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM*, Aug 2005, pp. 462–465.

[21] J. Kang, Y. Zhang, and J.-B. Ju, "Classifying DDoS attacks by hierarchical clustering based on similarity," in *2006 Internaltional Conference on Machine Learning and Cybernetics*, Aug. 2006, pp. 2712–2717.

[22] J. Xu and W. Lee, "Sustaining availability of web services under distributed denial of service attacks," *Transactions on Computers*, vol. 52, no. 2, pp. 195–208, Feb 2003.
[23] Y. Kim, W. C. Lau, M. C. Chuah, and H. J. Chao, "Packetscore: Statistics-based overload control against distributed denial-of-service attacks," in *INFOCOM*, 2004, pp. 2594–2604.
[24] A. Yaar, A. Perrig, and D. Song, "SIFF: A stateless internet flow filter to mitigate DDoS flooding attacks," in *IEEE Symposium on Security and Privacy*, May 2004, pp. 130–143.
[25] S. Jin and D. Yeung, "A Covariance Analysis Model for DDoS Attack Detection," in *Proc. of the IEEE International Conference on Communications (ICC)*, vol. 4, June 2004, pp. 1882–1886.
[26] C. Jin, H. Wang, and K. Shin, "Hop-Count Filtering: An Effective Defense Against Spoofed DoS Traffic," in *Proc. of the 10th ACM Conference on Computer and Communications Security*, 2003, pp. 30–41.
[27] A. Lakhina, M. Crovella, and C. Diot, "Characterization of network-wide anomalies in traffic flows," in *Proc. of the 4th ACM SIGCOMM Conference on Internet Measurement*, Italy, 2004, pp. 201–206.
[28] B. Krishnamurthy and J. Wang, "On network-aware clustering of web clients," in *Proceedings of the ACM SIGCOMM*, Stockholm, Sweden, Aug. 2000, pp. 97–110.
[29] M. Collins and M. K. Reiter, "An empirical analysis of target-resident dos filters," in *IEEE Symposium on Security and Privacy*, May 2004, pp. 103–114.
[30] M. Guirguis, A. Bestavros, I. Matta, and Y. Zhang, "Reduction of quality (RoQ) attacks on dynamic load balancers: Vulnerability assessment and design tradeoffs," in *Infocom'07: The IEEE International Conference on Computer Communication*, Anchorage, Alaska, May 2007, pp. 857–865.