An Unsupervised Approach to DDoS Attack Detection and Mitigation in Near-Real Time

Robert McAndrew
Colorado State University
rmcand@colostate.edu

Stephen Hayne
Colorado State University
stephen.hayne@colostate.edu

Haonan Wang
Colorado State University
wanghn@stat.colostate.edu

Abstract

We present an approach for Distributed Denial of Service (DDoS) attack detection and mitigation in near-real time. The adaptive unsupervised machine learning methodology is based on volumetric thresholding, Functional Principal Component Analysis, and K-means clustering (with tuning parameters for flexibility), which dissects the dataset into categories of outlier source IP addresses. A probabilistic risk assessment technique is used to assign "threat levels" to potential malicious actors. We use our approach to analyze a synthetic DDoS attack with ground truth, as well as the Network Time Protocol (NTP) amplification attack that occurred during January of 2014 at a large mountain-range university. We demonstrate the speed and capabilities of our technique through replay of the NTP attack. We show that we can detect and attenuate the DDoS within two minutes with significantly reduced volume throughout the six waves of the attack.

1. Introduction

Distributed Denial of Service (DDoS) attacks have received significant global attention, because they are increasing in frequency and severity [1]. DDoS occurs when attackers flood the target systems with huge amounts of traffic from many compromised systems, leading to interruption of the victim's services [2]. Direct costs to large organizations range from $50,000 to $100,000 per hour, and indirect costs can total much higher. We describe a system (NetBrane) designed to detect these DDoS in near real-time, and report on two different mitigation strategies based on our unsupervised outlier detection mechanism.

The contribution of this work lies in its combination of multiple features. First, our system is a near-real time monitor of all traffic on a network; that is, we can analyze traffic accurately without the need for sampling. This is a major benefit because using the entirety of the dataset for anomaly (outlier) detection provides the best possible results in terms of accuracy [25]. Next, our outlier detection mechanism is unsupervised, removing any dependence on having labeled data. It is impractical to obtain labeled data in many instances, especially in the case of a "new" attack whose profile is unknown. This freedom from labels also lets our mechanism be adaptive in the sense that it only seeks to identify behaviors that are "unusual" when compared to the majority of traffic. Such adaptivity allows for the potential to detect "new" attacks that supervised techniques cannot. Other domains are also coming to the conclusion that unsupervised learning is an attractive approach when dealing with unlabeled data [26, 27].

Detection and mitigation of DDoS is important because attackers increasingly use DDoS events as a smokescreen or distraction for more covert operations that allow them to carry out data breaches [3]. Our adversaries want not only to steal data or intellectual property (for later use or sale), but also to disrupt the operations of those targeted or impact their reputation. DDoS have been reported in the +1Tb/s range, driven by compromised Internet of Things (IoT) devices, such as digital video recorders and security cameras [4]. Trends in the size of DDoS appear stable; growing at approximately 6% per year since 2017 [1]. But the median size is erratic, with cyclic growth. It seems that when adversaries find new methods of attack, we see a new peak, followed by a decline when the method is mitigated (patched or blocked).

In 2019, 95% hit at 11.3Gbps or less. While tsunamis make headlines, the "small" ripples can still cripple a business. Our university was overwhelmed during a medium-sized DDoS in 2014 on two 10Gpbs connections to our Internet service provider (ISP).

In the last six months "the total number of attacks climbed by 84%, and the number of sustained (over 60 minutes) DDoS sessions doubled…extremely long attacks posted a massive 487% growth" [5]. Attackers have also resorted to small multi-vector attacks (using more than one service or attack type at a time). These "bit-and-piece" attacks beat detection thresholds because the targeted IP address receives only a relatively small number of responses in each organized campaign, leaving little or no trace. The typical ISP

URI: https://hdl.handle.net/10125/64534
978-0-9981331-3-3

response of blocking all traffic to an entire IP prefix cannot reasonably be applied; it is costly, due to blocking access to various legitimate services of many customers. We suggest that a finer grained detection and mitigation mechanism is required.

There are currently over 300 DDoS attack vectors, but the worst are those of "amplified reflection", where adversaries send relatively small queries to a server, spoofing a victim's IP address(es), and requesting a response involving a large amount of data. As a result, the server's and victim's network bandwidth will be flooded. It is the amplification factor/ratio of inbound to outbound data that makes the attack both easy and dangerous. Reflection attacks are occurring every 40 minutes, with the largest to date being 1.35Tbps using *memcached* UDP reflection (50000:1) [6].

In this paper, we study data captured from an actual NTP attack that occurred in 2014 on our campus with an amplification factor of 556, as well as a simulated attack in our network security lab. We conduct forensic re-analysis using our methodology to detect outliers in the flow data and apply the result to mitigate the effects of the actual DDoS in near real-time. Specifically, we detect unusual behaviors in two steps: (1) Functional Principal Component Analysis (FPCA) combined with (2) K-means clustering.

2. Related Work

Anomaly detection methods can be classified into (1) signature-based and (2) profile-based [7]. Signature-based methods use prior knowledge about characteristics of the anomaly of interest to identify suspects, and have several requirements, such as prior results from anomalies, the need for labeled data, and an external supervisor. Many machine learning classification techniques are "supervised", meaning that they need to be trained on a set of labeled data prior to use. Examples of popular approaches are the Support Vector Machine, Bayesian Networks, Neural Networks, and Discriminant Analysis (surveyed in [7, 8]). While these have been shown to perform well in certain situations where "known" anomaly data exists, the reliance on labeled data can be a difficult hurdle to overcome. For the case of network traffic classification, "ground truth" knowledge may not be available or even exist, thus supervised techniques can only be applied when the true labels are approximated. Training on incorrectly labeled data greatly skews results [9].

In the case of recent or new DDoS attacks, knowledge of which behaviors are malicious is not known; we do not have labels. Thus, supervised techniques cannot be applied. Profile-based methods create representative "normal" traffic behavior, and anomalies are detected by deviations from this profile. While there may be higher false alarm rates, profile-based methods are more promising due to their data-driven flexibility and they may also detect previously unknown anomalies [9]. Principal Component Analysis (PCA) is a widely used profile-based method which has been applied to detect traffic anomalies in DDoS data by decomposing network traffic into two components [24]. The anomalous subspace, which is noisier and contains the significant traffic spikes, is separated from the normal, which is dominated by predictable traffic. An individual observation is deemed an anomaly if its projection to the anomalous subspace is large. A two-stage approach was proposed, using (1) PCA to identify potential anomalies, and (2) a meta-heuristic to group them [10].

However, the use of PCA has been criticized due to issues pertaining to (i) false positive rates, (ii) traffic measurement aggregation, (iii) normal subspace pollution and (iv) correct anomaly identification [11]. The third is important, as it highlights the need to choose which principal components represent "normal" behavior, and which ones represent the "abnormal". It has been demonstrated that some traffic captures do not lend themselves to this partition/selection; that is, all principal components contain abnormal behaviors, and thus this approach is not usable [28].

Clustering is another example of a profile-based method. Clustering has been applied to all traffic, comparing the centers of known "normal" traffic clusters to the centers of actual traffic, to try and determine if the actual traffic is not normal [12]. Unfortunately, this specific approach has only been applied to Simple Network Management Protocol (SNMP) objects, not network flows, and requires known normal traffic data. Clustering techniques have been used to characterize DDoS attack traffic (K-means, Clustering Large Applications (CLARA), and Self Organizing Maps) [13]. K-means was found to be the most accurate for attack detection because attack traffic has strong similarity as opposed to the heterogeneity of normal traffic. In this research, "attack" clusters still mixed in legitimate traffic with malicious (between .4% and 2.04%). We believe this phenomenon can be eliminated by clustering only demonstrated "outliers", not all traffic.

To avoid the issues we have identified with PCA and clustering when applied separately, we will use FPCA (instead of PCA) and apply clustering to the resulting outliers [28] (that paper examined "scanner" behavior, where here we analyze a DDoS attack). We perform classification only using the data that is given as input, making this technique well-suited for dealing with an unknown attack. We suggest this is more appropriate than using a supervised approach trained

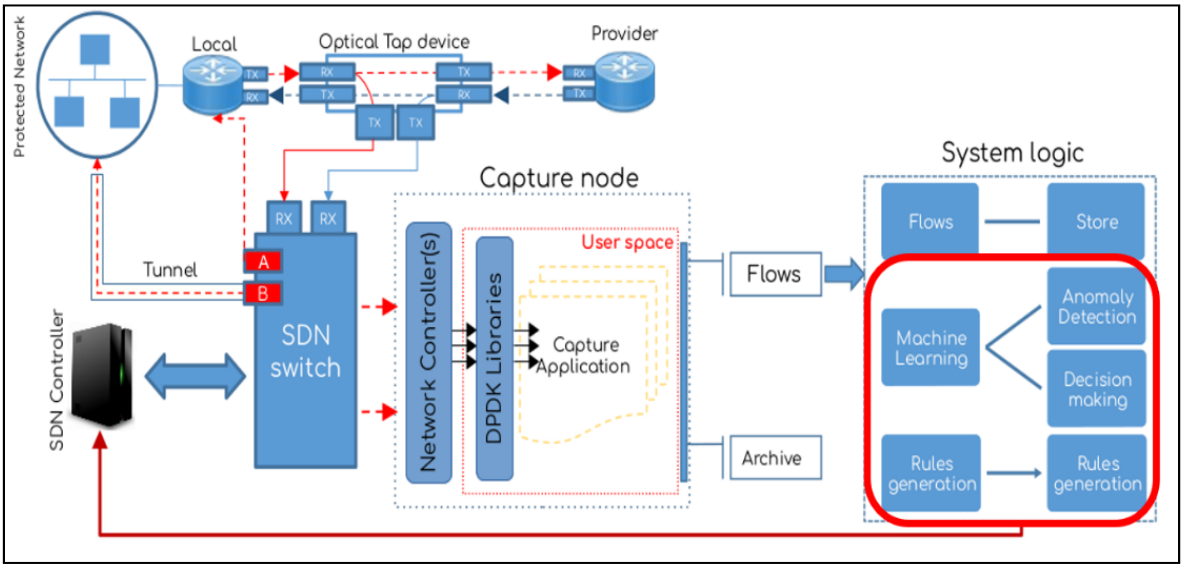

Figure 1. NetBrane system architecture

on data from a previous attack, as there are a wide variety of different attack vectors, and what was previously learned may not apply.

When ground truth knowledge of true perpetrators in an attack is non-existent, the notion of "false-positives" in anomaly detection arises. Frequently, these are potentially controlled with risk assessment, computational trust, and reputation models (for a survey, see [14]). Methods based on probabilistic risk assessments are widely used and seem to provide promising results [15]. We introduce a probabilistic approach to risk assessment which assigns a "threat level" to potential attackers.

3. System Design and Dataset Description

The size of the organization does not matter when it comes to protection from attack. Big, small, startup: hackers still want your data and they will stealthily poke holes in your network to find the access points. While "security as a service" (SECaaS) exists (e.g., Qualys, Sentinel, Sophos, Proofpoint, along with offerings from the major cloud computing companies such as AWS, Azure, Google) and can offer some protection, current solutions cannot benefit everyone; SECaaS is usually cloud-based without requiring any on-premise hardware or much software distribution. However, many organizations, such as government, military, and financial organizations, need to tightly control their data which is incompatible with SECaaS – (meta) data cannot be shipped off-premises.

To bridge this gap, we have built a system called *"NetBrane"* (network membrane, [29]). NetBrane is a defense service where technologies are combined to construct a shield while leaving data and sensitive services on the premises. Figure 1 shows the NetBrane architecture. Key novelties of the project lies in the confluence of: (a) Software-Defined Networking (SDN) enabled small distributed footprint with 100G capture/filter capability for neutralizing DDoS (left side of figure), (b) elastic data analytics using near-real time flows and cloud capabilities (all analytics described in this paper are conducted in the section enclosed within the red box), (c) situational awareness, in terms of the global Internet information, and (d) proactive reconnaissance, by intelligent synthesis of information from multiple sources. The design calls for NetBrane nodes to reside in points-of-presence (POPs), capturing and summarizing traffic at line speed, finding anomalies worthy of creating filter rules for, pushing these filters to the local SDN infrastructure, communicating with the appropriate POPs routing infrastructure to block traffic, while tunneling legitimate traffic to its destination. We use SDN because it allows for dynamic software control of network design and operations. Unfortunately we have discovered that openflow will not function at high line rates (>20gbps), and have had to design and implement a system called FlowRide (not described here).

At our large mountain-west university, we have installed optical taps to capture network flows (top left, Figure 1) at line rate. FlowRide pushes those flows into message queues, which are read by our analytics engine (red box on right side of Figure 1) in near-real time where traffic is characterized (scanner or attack detection); data is saved in parallel to hadoop (HDFS) for data lake analytics. We read these flows from the message queue in small time intervals and analyze them, applying multi-core (parallel R packages). We have currently demonstrated resilience up to 400Gb/s.

The real-world raw data we consider in this paper is a collection of bi-directional flow records to and from

our university, relating to the NTP service. We focus on traffic between January 12 and January 25 of 2014, during the second half of which a true real-world amplified reflection DDoS was carried out (starting in the early morning of January 18). This attack impacted the university in six waves (see Figure 4 for a plot of packet counts), with a wave defined by significantly decreased packet volume, or the monitoring system becoming unavailable.

The flow records contain timestamp, source and destination IP (SIP & DIP), source (SRC) and destination (DST) port, packet and byte counts. We currently only analyze TCP data; we plan to consider UDP in future work. We group information into one-minute bins, and the full dataset covers roughly twenty-thousand minutes. As this is a real-world dataset, we lack "ground truth" knowledge of which SIPs are the victims (spoofed by attackers). However, we suggest that ground truth is not necessary as we know that an amplified reflection DDoS occurred, and we only seek ways to alleviate damage.

The synthetic data we consider is very similar to the NTP attack data in terms of flow records. This data is grouped into one-minute bins, but the total dataset only covers forty minutes. This attack comes in one wave, for which we do have ground truth. There are twenty true attackers, all with SIPs of the form 10.1.7.X, targeting one victim with SIP 129.82.138.136 on port 80. These attackers send approximately 20 million packets during the attack.

4. Methodology

Upon initialization of our analytics system, we aggregate the most recent thirty-minutes of Internet traffic (packet and byte count separately) into one-minute bins. For this initial thirty-minute window, we assume that we are not under an attack and have relatively "usual" traffic. When one minute has passed, we "slide" this window to cover the new minute's worth of data and drop the first observation from the previous window (i.e., the "oldest" minute of data). With this mechanism, we always have the most recent thirty-minute time series of traffic volumes, allowing us to monitor for potential attacks in near-real time.

In each iteration of the thirty-minute window, two thresholds (one for packets and one for bytes) are calculated and used for volumetric attack detection. The threshold is given by Equation (1),

$$Thresh = \max\{X_t \mid t \in H\} + cv \cdot \mathrm{SE}[\max\{X_t \mid t \in H\}]. \quad (1)$$

In the above equation, X_t for $t \in H$ is the time series of packets or bytes in the given window of history. $\mathrm{SE}[\max\{X_t \mid t \in H\}]$ is the standard error of the maximum packet or byte count from a LOESS fit of the packet/byte time series in the window of history. Lastly, *cv* is a critical value determined from investigation of long-term (months) packet and byte distributions.

When our window slides and gathers the new, most recent minute of packet and byte counts, these values are compared to the thresholds calculated in the previous window iteration. That is, we check if the new packet/byte count exceeds their respective thresholds. If they are below their thresholds, the thresholds are recalculated, and the process is repeated when a new minute's worth of data is collected. If at least one of the counts exceeds their threshold, we believe a DDoS has been detected, and begin our attack mitigation.

The motivation for this threshold is as follows: when not under attack, previous "large" volumes and counts are considered acceptable, so we believe we are under attack from a DDoS when new data exceeds the largest value in the window of history by more than a scaled measure of the maximum's variability. This also captures the idea that we may see "normal" network activity that is larger than a previously accepted amount, but only see potential for a DDoS if new packet or byte counts exceed what we expect from historical variability of our data.

When an attack is first detected, our system decides which destination port the attack is being launched on. This port is chosen based on the largest relative change in the minute at which the attack was detected. That is, the system has already noticed a large increase in the traffic when aggregated across all ports, so we now focus on the specific port that saw the largest increase. We refer to this as the "attack port", and then attempt to identify attacker SIPs on that port.

For each SIP that contacted a DIP on the attack port in the given window, we construct a time series (by minute) of packet counts sent and received by that SIP. These time series are then used as input for Functional Principal Component Analysis (FPCA), and outliers are determined using the FPCA scores. We perform a "two-pass" implementation of FPCA; that is, after identifying outliers from one application, they are removed from the dataset and FPCA is re-run to flag additional outliers. This portion of the analytics is described in more detail in Section 4.1. Once outlier SIPs – the potential attackers – are gathered, risk assessment is carried out and a threat level is assigned to each. This threat level exists between 0 and 1 and is intended to represent the likelihood of a SIP being an attacker, with a value closer to 1 indicating malicious activity. This risk assessment is described in more detail in Section 4.2.

After the first minute of attack analytics, we switch our "sliding" window to one that is a "growing" window. That is, we do not drop the oldest observation when a new minute is gathered. This is done so that we do not only investigate attack volumes when mitigating the attack. Note that when the attack is first detected, we have 29 minutes of "usual" traffic, and FPCA finds outliers by identifying significant differences between SIPs in this period of "usual" activity and the attackers. If the sliding window is used and the attack continues for a large amount of time, we may eventually encounter an instance where "usual" activity is drowned out by the attackers, or is non-existent, which hinders the ability of FPCA to find all significantly different SIPs. With each new minute, the outlier detection procedure and threat level assignment are repeated.

We then perform DDoS mitigation using the set of outliers found by our system. For the SIPs flagged to be an outlier by FPCA in previous iterations of attack analytics, firewall rules are created to block their traffic from the network in future minutes. That is, the SIPs with unusual traffic volumes are prevented from impacting the network any further, dampening their effect on the system. In addition, previously determined outliers are not considered in subsequent FPCA analyses, so new potential attackers can be identified and blocked, leading to continued mitigation.

To determine if an attack has stopped (or significantly declined), we set a limit on how long we expect to see traffic return to "usual" levels. When new minutes' data stay below the thresholds that were initially exceeded for one hour, we think that we are no longer under attack. At this point, the analytics system removes all outlier SIPs from being blocked and returns to calculating the packet/byte thresholds until another attack is detected. In addition, the "growing" window reverts back to a "sliding" window, snapping to the most recent thirty minutes of traffic. One hour is chosen because it is double the size of our sliding window. That is, we revert to monitoring the traffic rather than mitigating it when we are sure that volumes have returned to "usual" levels, and our thresholds will not be inflated by including attack traffic.

4.1. FPCA + K-means

The procedure begins with application of FPCA in order to first classify "outliers" in the data. We construct an $n \times T$ matrix whose (i,t) entry is the count of packets sent and received by the ith SIP during the tth minute. FPCA models this as a mean series plus a linear combination of eigenfunctions, which are orthogonal curves representing the descending dimensions of variance in the data; that is, the first eigenfunction can be thought of as the direction of highest variability, eigenfunction two the second most variable, and so on. We employ the Principal Analysis by Conditional Expectation (PACE) algorithm of [16]. In order to select the number of eigenfunctions in our model, we apply the Akaike information criterion (AIC) and the Bayesian information criterion (BIC) [17]. For the data presented here, these agree on parameter selection; but we acknowledge this may not always be the case. Context-specific factors should be considered when deciding which criterion is more appropriate [18].

To classify SIPs, we calculate each observed series' FPCA scores, which are projections of the data onto the eigenfunctions. Each SIP has one score for every eigenfunction, and that SIP is flagged as an "outlier" if at least one of its scores exceeds a three standard deviation threshold from the mean (well-known due to its standard application based on Chebyshev's inequality [19]). For example, from the n scores on the first eigenfunction, we can calculate the bounds $xbar \pm 3s$; $xbar$ is the mean score and s is the standard deviation. Any SIP whose first eigenfunction score lies beyond these bounds is flagged as an outlier. We use the term "outlier" because we do not think all SIPs flagged by FPCA are attackers - these are SIPs that contacted the network in an unusual way, which can clearly include other activity. Because of this, we carry out the second step of clustering these abnormal SIPs based on their rate of successful connections, where a "success" is characterized as the DIPs sending at least one packet back to the SIP. With this, we can investigate the cluster that exhibits behavior expected of an attacker, as our abnormal SIPs are now separated by their connectivity with the network.

In order to perform this clustering, we employ the K-means algorithm of [20]. The number of clusters in the application of K-means is chosen with the "elbow method", which seeks the cluster amount such that adding one additional cluster would not have a significant impact on the fraction of variance explained (FVE) in the entire dataset [21]. K-means is run multiple times using randomly generated centers in order to assess sensitivity with respect to their centers, and we find that our data does not exhibit sensitivity to center selection.

4.2. False-Positives and Risk Assessment

In each iteration of our "growing" window when under attack, a set of outlier SIPs is collected as potential attackers. We do not suppose that all outliers are attackers, so we aim to introduce a quantitative mechanism to allow an operator to filter out possible false positives (non-attackers identified as outliers).

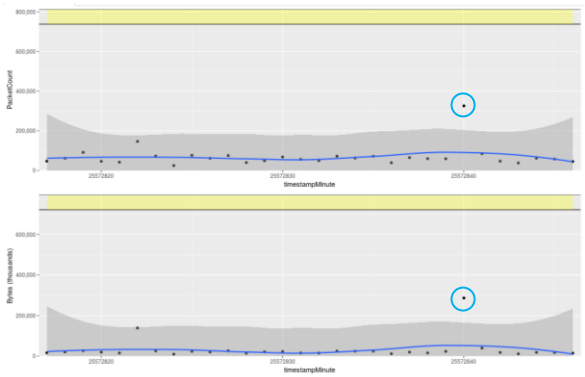

Figure 2. Initialization, simulated attack

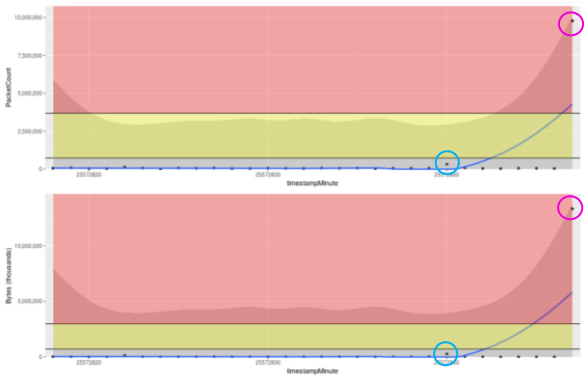

Figure 3. Simulated attack detected

We call this mechanism a "threat level", which is a value between 0 and 1, with a value closer to 0 indicating a higher likelihood of a false-positive (non-threatening).

To calculate this threat level, we first gather the total data sent and received by each outlier SIP and use these to construct a cumulative density estimate of "outlier" data. Then, we take a sample of size 200 (or as many as we possibly can, should there be less than 200) from the non-outlier SIPs, and construct a similar cumulative density estimate from their total volumes sent and received. This gives us two cumulative density estimates: one for the outliers, and one for the non-outliers.

Next, for each outlier SIP we calculate its percentile in both cumulative density estimates. That is, each outlier SIP has a corresponding *p1*, which is the probability that an outlier has volume less than or equal to that of the given SIP, and *p2*, which is the probability that a non-outlier has volume less than or equal to that of the given SIP. The threat level is then calculated by Equation (2),

$$\textit{Threat Level} = \min(1,\max(0, p1 - (1 - p2))). \qquad (2)$$

This threat level is motivated by using the SIPs not labeled as outliers to determine if the outliers found are false positives. If an outlier's volume is low, it will be closer to the distribution of non-outliers, making us think that it is a false-positive. For example, suppose an outlier SIP is a false-positive (non-attacker). Then, the location of that SIP in the outlier cumulative density estimate will be close to the body of the non-outlier cumulative density estimate, making *p1* low and *p2* high. This translates into a threat level close to zero. Compare this to the case where we have an outlier SIP that is an attacker. This SIP will have both *p1* and *p2* large, translating into a larger threat level.

With each outlier being assigned a threat level, operators can be more measured in their "blocking" during an attack. If an outlier with a low threat level is a known or acceptable SIP, then it may not need to be blocked from the network. This decision would require specific knowledge of the network and we leave this decision to operators at this time. In our analyses for this paper we always block all outliers collected.

5. Results

We apply our attack detection and mitigation methodology to a simulated DDoS attack as well as an amplified reflective DDoS attack from 2014. The simulated attack is discussed first, with focus on attack detection. Ground truth from this event allows us to use this application of our methodology as validation. Following the synthetic event, we discuss the real-world NTP attack with focus on attack mitigation.

5.1. Simulated Attack

Our system first initializes on a thirty-minute window in which we are not under an attack. The packet (top) and byte (bottom) count time series are shown in Figure 2, with the line separating the yellow region above indicating the thresholds for attack detection, as calculated in this window. Note the observation circled in teal – this is the largest value in our window. At this point, there is a mean packet count of 50 thousand and a mean byte count of 25 million. After initialization, the simulated attack was started, so the next minute of data will include attack traffic. When the new data is received by the system, the aggregated packet and byte counts are compared to the previous thresholds. Figure 3 shows both thresholds being exceeded (by the point circled in pink), which indicates that our system is under attack (also denoted by the red region above the yellow). Notice that the point circled in teal is the same value circled in Figure 2, showing the scale of this attack. We now know the simulated attack has begun, validating correct attack detection within its first

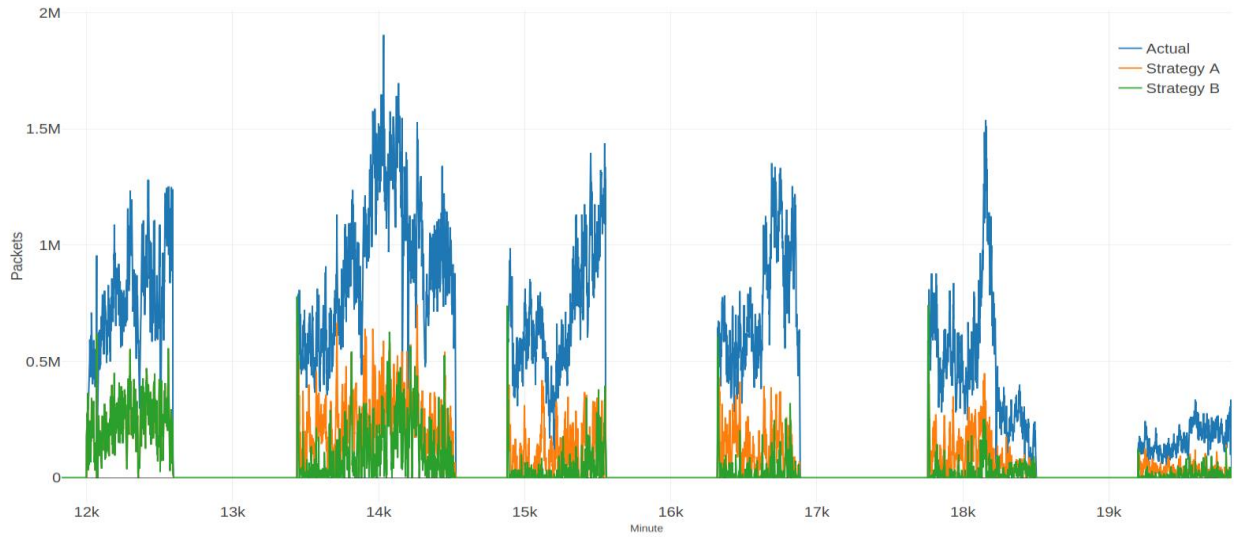

Figure 4. NTP attack packet counts - actual (blue), "strategy a" reduced (orange), "strategy b" reduced (green)

minute. With this new minute of data, the average packet count rises to 4.4 million, and the average byte count rises to 6.25 billion.

Next, we seek to determine the attack port by finding the port that had the largest relative increase in the most recent minute. In this case, this is identified to be port 80, which is the actual destination port being targeted by the simulated attack.

Table 1 shows the outlier SIPs along with their cluster center from application of K-means and threat level. Note that all the attackers belong to the same cluster with low proportion of successful contacts, while the victim is alone in the cluster with a high proportion. This separation is due to the victim appearing in the period of "usual" activity prior to the attack. It was behaving in its usual way, reaching out to other IPs on the network and receiving responses. The attackers do not appear in this portion of the dataset, only coming into play during the most recent minute of the window. They only contact the victim, and since they are performing a DDoS and sending large volumes, they receive no responses.

We also see a separation between attackers and victim in the form of the threat level. All attackers have threat levels of at least 88%, which is appropriate because we want a larger threat level to indicate malicious SIPs. The victim has a threat level of about 5%, which accurately reflects the fact that it is a false-positive (non-attacker outlier). We believe it is quite useful that this technique captures the victim because it likely removes a secondary step of further investigating the attackers to determine their target.

When the next minute of data is gathered, we block traffic from all twenty-one of these outliers. This significantly reduces the volume seen on the network, and returns packet and byte counts to below their respective thresholds. This minute still involves attack traffic, but we have mitigated all of it since we have identified all malicious SIPs. The same is true for the remaining minutes of the dataset – we stay below our thresholds and mitigate the DDoS event. This analysis focuses on an attack that only comes in one wave and does not have enough "usual" traffic following the simulated event to fully discuss when to stop blocking the identified outliers from the network.

Table 1. Simulated DDoS attack - outlier summary

SIP	Cluster Center	Threat Level
10.1.7.133	0	0.9961
10.1.7.141	0	0.9903
10.1.7.89	0	0.9785
10.1.7.150	0	0.9779
10.1.7.136	0	0.967
10.1.7.20	0	0.967
10.1.7.37	0	0.9554
10.1.7.113	0	0.9533
10.1.7.53	0	0.9327
10.1.7.147	0	0.9286
10.1.7.23	0	0.9272
10.1.7.85	0	0.9259
10.1.7.127	0	0.9203
10.1.7.71	0	0.9189
10.1.7.134	0	0.9148
10.1.7.148	0	0.9134
10.1.7.81	0	0.9108
10.1.7.82	0	0.9033
10.1.7.58	0	0.8978
10.1.7.149	0	0.8801
129.82.138.136	0.94	0.0527

5.2. NTP Attack

In applying our detection mechanism to the real-world NTP amplified reflection DDoS attack, the packet threshold is immediately exceeded in the first minute of the first wave of the attack. As this dataset consists only of NTP traffic, the step of determining the "attack port" is unnecessary.

Recall our mitigation strategy: we "remember" the SIPs we flag as outliers and block their activity until aggregate traffic stays below our thresholds for one hour. This attack includes six waves, each of which is more than one hour after the end of the previous (as shown in Figure 4). Because of the big gaps between the waves, our analytics system treats these waves as six different attacks, as we "forget" outlier SIPs from previous waves. Since we now have the knowledge that this is one attack, we aim to compare our current strategy to one in which outliers are not forgotten and their traffic continues to be blocked. For the purposes of this discussion, we refer to the strategy of forgetting outliers after one hour of usual activity as "Strategy A", and the alternative of never forgetting outliers as "Strategy B".

Figure 4 shows the time series (by minute) of packet counts sent and received on the network for the NTP DDoS event. The actual packet count series of the event is shown in blue. The series shown in orange is the remaining packet counts after mitigation Strategy A has been applied, and the series shown in green is after mitigation Strategy B has been applied; that is, these are the packet counts that would have been seen if our blocking rules had been in effect (actual packet count minus outliers' packet count).

First observe in Figure 4 that the series of packet counts for both Strategy A and Strategy B are well below that of the actual attack. This visually indicates that our mitigation procedure is effective in reducing the impact of the attack. Numerically, we can investigate total packet counts across the entire attack for all three series. In the actual attack, approximately $2.8 \cdot 10^9$ packets were sent and received across the network on the NTP service. Applying Strategy A brings the total packet count to approximately $6.4 \cdot 10^8$, or 23% of the true attack (a 77% reduction in packets). Applying Strategy B brings the total packet count to approximately $2.4 \cdot 10^8$, or 8.7% of the true attack (a 91.3% reduction in packets).

To more formally compare the packet counts of the attack and our two mitigation strategies, we perform paired t-tests for each of the three combinations [22]. That is, we test for significant differences between the packet counts of the attack and Strategy A, the attack and Strategy B, and both strategies. In all three of these tests, a p-value of less than $2 \cdot 10^{-16}$ is reported, indicating strong statistical evidence for a difference in these time series. From the visual inspection of Figure 4, we certainly expected the reduced packet count series to be different from the true attack, but we also see a significant difference between Strategy A and B. To further investigate their difference, we calculate the Dynamic Time Warping (DTW) "distance" between the two packet counts – a smaller distance implies a greater similarity in the series [23]. The DTW "distance" is calculated to be approximately $2.7 \cdot 10^8$. While this seems large, it is relatively small when compared to the DTW "distance" between the true attack and the two strategies: Strategy A is roughly $2.5 \cdot 10^9$ away from the full un-mitigated attack, and Strategy B is at almost $3.5 \cdot 10^9$. We expected Strategy B to be further from the true attack because of the larger packet reduction it achieved, but it is interesting that we observe such a significant difference between the resulting time series of Strategy A and B. Strategy B clearly outperforms Strategy A. Further discussion about these two strategies is included in Section 6.

This mitigation includes the steps of outlier detection, clustering, and threat level assignment in our analytics. Recall that, for this analysis, we "remember" and block all outliers found in future traffic. This makes the resulting clusters and threat levels calculated throughout the NTP attack independent of our mitigation. This does not always need to be the case, as the system (or an operator) could block traffic from only outliers with a threat level above a specified threshold, outliers in certain clusters, or a combination of the two. In any instance of this, fewer outliers would be blocked than were found, and the mitigation achieved would not be as large as that from Strategy A or Strategy B. That is, the mitigation we are comparing here is between extremes – the true attack and blocking all outliers. As such, we do not investigate the effect of blocking subsets of outliers in this paper. For our analysis, the clusters and threat levels were used to better understand the types of behaviors that were apparent during the attack. This is a benefit that was highlighted in the smaller simulated attack of Section 5.1, and one that an operator would be able to use as well.

6. Discussion

Our attack detection mechanism relies on the sliding-window approximation of real-time streaming data. Thirty-minutes is selected as the window size because it is a near "worst-case" scenario in terms of how much data we need for our statistical procedures to be applicable. We want our FPCA results to be accurate and stable, and we feel going below thirty observations for each series would breach this. A larger

window could be kept for attack detection, but this would impact the thresholds in each iteration of the window. This would also alter the set of outliers found when an attack is detected, as we would have more "usual" activity in the beginning of the time series.

In Section 5.1, we applied our attack detection methodology to a simulated DDoS attack. Our analytics detected the attack within its first minute of activity, and accurately identified the twenty attackers as well as the one victim. The clustering results and threat level assignments clearly separated attackers from victim. In the real-world attack of Section 5.2, we cannot expect to see such a distinct stratification of outliers because we do not have "ground truth" knowledge of the attackers. We cannot check if they have significantly larger threat levels or appear in clusters distinct from the non-attackers. Further, we do not know how many true attackers there are, so the threat level procedure we implement might produce "deflated" values for truly malicious SIPs.

To demonstrate this, consider the early phases of an attack with many malicious IPs. Imagine our unsupervised FPCA outlier identification produces a set of SIPs that is only a subset of the attackers (because some have not had enough time to fully behave like an attacker), so that some attackers are left in the non-outlier set. In assigning threat levels, we compare probabilities from outlier and non-outlier cumulative density estimates, and having attackers included in the non-outlier set makes the non-outlier distribution closer to that of the outliers. In turn, the approach think that outliers are more like "usual traffic", producing a lower threat level. Note that we attempt to reduce the impact of this issue by creating non-outlier cumulative density estimates from a sample of non-outliers, so it is possible that we will avoid attackers that have not yet been flagged as outliers. Even with this, we concede that it is possible for some attackers to be treated as non-outliers – this is very difficult to control for without ground truth knowledge of the dataset.

We compared two mitigation strategies in Section 5.2 – Strategy A involved "forgetting" outliers and resetting blocking rules when an attack subsides below initial thresholds for an hour while Strategy B mimicked a perfect memory and continual blocking. Strategy A reduced total packet counts of the event to 23% of the original un-mitigated amount, and Strategy B reduced it to 8.7%. Strategy B achieves greater packet reduction, as it immediately blocks SIPs that were flagged as outliers in previous waves. We suggest Strategy B is most useful when a "botnet" is being used for an attack, because the IPs are "re-engaging" after a pause. By building up this botnet list, and completely blocking them, they cannot even "restart" the attack. Further, this is why the reduced packet counts are identical in the first wave shown in Figure 4 (the green and orange series over plot) – there are no previous outliers for Strategy B to block.

Note that mitigation achieved is not the only difference between these Strategies. In all waves after the first, Strategy A allows traffic through that was previously being blocked, increasing the packet counts relative to Strategy B, while also providing a different set of SIPs for FPCA to use as input. As a result, this also changes the threat level calculation, and introduces a greater chance for having attackers in the non-outlier set.

This may seem to indicate that a "perfect memory" of outliers after an attack has been detected is superior, but this does not account for the true nature of the real world. During an actual DDoS attack, there is no way to tell how many "waves" there will be and when they will stop. Due to this, we initially recommend "forgetting" the outliers and returning to a sliding window (monitoring for the start of an attack) after one hour. We allow for human operators to interact and configure the system to implement Strategy B for a while, and then to reset when ready.

We do not suggest that our methodology can be used as a "set-and-forget" piece of software, but rather a strong supplemental tool to an operator or operating team. Consider our mechanism detecting the start of a DDoS attack and informing humans. Outlier SIPs will be blocked, mitigating the attack, while summary information (clusters and threat levels) are provided to operators every minute. They then have at least one hour to investigate further and more accurately determine the nature of the attack. For example, suppose a "false alarm" is detected (large packet/byte counts that do not truly represent an attack). If the operator determines that this was a false alarm, they can stop the blocking and not have to wait an hour the system to return to attack monitoring. Alternatively, should a true attack be detected, and operators think there may be waves, the one-hour limit can be removed so that larger and faster mitigation is achieved. In all, we suggest the length of time it takes for the analytics system goes from attack mitigation back to detection is a tuning parameter that should be informed by specific knowledge of the network/institution.

7. Conclusions

We have demonstrated an unsupervised, adaptive technique for detecting and mitigating DDoS attacks on both synthetic and real-world datasets. Dynamic thresholding is shown to detect the attack, and the FPCA+Kmeans approach mitigates the volume significantly (by more than 90%). Such unsupervised

approaches are best suited for detection and mitigation of "unknown" attacks. We have investigated two strategies for reducing packet and byte counts during an attack and suggest operators with network-specific knowledge can use both as appropriate. Assignment of probabilistic threat levels to the outliers allows for better understanding of the SIPs identified.

Acknowledgements

This material is based on research sponsored by the Department of Homeland Security (DHS) Science and Technology Directorate, Homeland Security Advanced Research Projects Agency (HSARPA), Cyber Security Division (DHS S&T/HSARPA CSD) BAA HSHQDC-14-R-B0005, and the Government of United Kingdom of Great Britain and Northern Ireland via contract number D15PC00205. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Department of Homeland Security, the U.S. Government, or the Government of United Kingdom of Great Britain and Northern Ireland. The research of Dr. Wang was partially supported by NSF grants DMS-1737795 and DMS-1923142.

References

[1] Akamai, "The State of the Internet Report," 2019. https://www.akamai.com/us/en/resources/our-thinking/state-of-the-internet-report/.

[2] Kaspersky, "Kaspersky Labs," 2018. https://usa.kaspersky.com/.

[3] S. Mansfield-Devine, "The Growth and Evolution of DDoS," *Network Security*, pp. 13-20, 2015.

[4] B. Krebs, "KrebsOnSecurity Hit with Record DDoS," 2016. https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/.

[5] O. Kupreev, E. Badovskaya and A. Gutnikov, "DDoS Attacks in Q1 2019," https://securelist.com/ddos-report-q1-2019/90792/.

[6] Cloudflare, "NTP Amplification Attack," 2019. https://www.cloudflare.com/learning/ddos/ntp-amplification-ddos-attack/.

[7] M. Ahmed, A. N. Mahmood and J. Hu, "A Survey of Network Anomaly Detection Techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19-31, 2016.

[8] A. Singh, N. Thakur and A. Sharma, "A review of supervised machine learning algorithms," in *Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference on*, New Delhi, 2016.

[9] M. Soysal and E. G. Schmidt, "Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison," *Performance Evaluation*, vol. 67, pp. 451-467, 2010.

[10] G. Fernandes, L. F. Carvalho, J. J. Rodrigues and M. L. Proenca, "Network anomaly detection using IP flows with principal component analysis and ant colony optimization," *Journal of Network and Computer Applications*, vol. 64, pp. 1-11, 2016.

[11] H. Ringberg, A. Soule, J. Rexford and C. Diot, "Sensitivity of PCA for traffic anomaly detection," *ACM SIGMETRICS Performance Evaluation Review*, vol. 35, pp. 109-120, 2007.

[12] W. Cerroni, G. Monti, G. Moro and M. Ramilli, "Network attack detection based on peer-to-peer clustering of SNMP data," in *International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, Houston, 2009.

[13] B. Hammi, M. C. Rahal and R. Khatoun, "Clustering methods comparison: Application to source based detection of botclouds," in *Security of Smart Cities, Industrial Control System and Communications, 2016 Intntl. Conf. on*, Paris, 2016.

[14] D. D. S. Braga, M. Niemann, B. Hellingrath and F. B. D. L. Neto, "Survey on Computational Trust and Reputation Models," *ACM Computing Surveys (CSUR)*, vol. 51, p. 101, 2018.

[15] Y. Cherdantseva, P. Burnap, A. Blyth, P. Eden, K. Jones, H. Soulsby and K. Stoddart, "A review of cyber security risk assessment methods for SCADA systems," *Computers & security*, vol. 56, pp. 1-27, 2016.

[16] F. Yao, H.-G. Muller and J.-L. Wang, "Functional data analysis for sparse longitudinal data," *Jrnl. of the American Statistical Association*, vol. 100, pp. 577-590, 2005.

[17] S. I. Vrieze, "Model selection and psychological theory: a discussion of the differences between the AIC and the BIC," *Psych. methods*, vol. 17, p. 228, 2012.

[18] Y. Li, N. Wang and R. J. Carroll, "Selecting the number of principal components in functional data," *Jrnl. of the American Statistical Association*, vol. 108, pp. 1284-1294, 2013.

[19] S. Seo, *A review and comparison of methods for detecting outliers in univariate data sets*, Pittsburgh, 2006.

[20] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *Jrnl. of the Royal Statistical Society, C*, vol. 28, pp. 100-108, 1979.

[21] D. J. Ketchen and C. L. Shook, "The application of cluster analysis in strategic management research: an analysis and critique," *Strategic Management Journal*, vol. 17, pp. 441-458, 1996.

[22] H. Hsu and P. A. Lachenbruch, "Paired t Test," *Wiley encyclopedia of clinical trials*, pp. 1-3, 2007.

[23] P. Tormene, T. Giorgino, S. Quaglini and M. Stefanelli, "Matching incomplete time series with dynamic time warping: an algorithm and an application to post-stroke rehabilitation," *Artificial intelligence in medicine*, vol. 45, pp. 11-34, 2009.

[24] A. Lakhina, M. Crovella and C. Diot, "Diagnosing network-wide traffic anomalies," in *ACM SIGCOMM Computer Communication Review*, New York, 2004.

[25] J. Mai, C.-N. Chuah, A. Sridharan, T. Ye and H. Zang, "Is Sampled Data Sufficient for Anomaly Detection?," in *6th ACM SIGCOMM conference on Internet measurement*, 2006.

[26] T. Tlusty, G. Amit and R. Ben-Ari, "Unsupervised clustering of mammograms for outlier detection and breast density estimation," in *24th International Conference on Pattern Recognition*, 2018.

[27] F. Carcillo, Y.-A. Le Borgne, O. Caelen, Y. Kessaci, F. Oble and G. Bontempi, "Combining unsupervised and supervised learning in credit card fraud detection," *Information Sciences*, 2019.

[28] R. McAndrew, M. Gharaibeh, H. Wang, S. Hayne and C. Papadopoulos, "A Functional Approach to Scanner Detection," in *Proceedings of the Asian Internet Engineering Conference*, Bangkok, 2017.

[29] "NetBrane, Funded Project, Department of Homeland Security Award D15PC00205," 2015-2019.