# AMON-SENSS: Scalable and Accurate Detection of Volumetric DDoS Attacks at ISPs

Rajat Tandon*, Pithayuth Charnsethikul*, Michalis Kallitsis†, and Jelena Mirkovic*
*University of Southern California Information Sciences Institute, Marina Del Rey, CA, USA
†Merit Network Inc., Ann Arbor, MI, USA

***Abstract*—Distributed Denial of Service attacks continue to be a severe threat to the Internet, and have been evolving both in traffic volume and in sophistication. While many attack detection approaches exist, few of them provide easily interpretable and actionable network-level signatures. Further, most tools are either not scalable or are prohibitively expensive, and thus are not broadly available to the network operator community.**

**We bridge this gap by proposing AMON-SENSS, an open-source system for scalable, accurate DDoS detection and signature generation in large networks. AMON-SENSS employs hash-based binning with multiple bin layers for scalability, observes traffic at multiple granularities, and deploys traffic volume and traffic asymmetry change-point detection techniques to identify attacks. It proactively devises network-level attack signatures, which can be used to filter attack traffic. We evaluate AMON-SENSS against two commercial defense systems, using 37 days of real traffic from a mid-size Internet Service Provider (ISP). We find that our proposed approach exhibits superior performance in terms of accuracy, detection time and network signature quality over commercial alternatives. AMON-SENSS is deployable today, it is free, and requires no hardware or routing changes.**

## I. Introduction

Distributed denial-of-service (DDoS) attacks are increasing in volume and frequency [1], [2]. A recent report from NetScout [3] reveals that more than 5 M attacks occurred between January and June 2021, including multiple attacks exceeding 1 Tbps. Many networks today handle volumetric DDoS attacks by purchasing either in-network commercial defense appliances, or by subscribing to cloud-based DDoS defense services. In both cases, the commercial defense profiles the traffic to the potential targets and derives a *network signature*, specifying the target and traffic ports and protocols. The defense uses this signature to filter attacks. A commercial defense can also resort to deep packet inspection or other more sophisticated traffic analyses when filtering traffic, to minimize collateral damage. Commercial defenses have two downsides: cost and lack of transparency. While clouds offer basic business plans relatively cheaply (e.g., $200 per month for CloudFlare, $3,000 per year for AWS Shield), the price quickly increases for larger networks or networks carrying larger traffic volumes. Commercial appliance costs range from tens to hundreds of thousands of dollars. Commercial defenses further employ proprietary algorithms to devise filtering rules, and there are no independent evaluations of their accuracy.

In many cases volumetric DDoS attacks could be handled by the target network or the target's upstream ISP, foregoing the cost and uncertain performance of commercial defenses. ISPs are especially well positioned for this task – they already handle high traffic volumes, they are rarely overwhelmed by DDoS attacks launched at their customers, and they have close relationships with their customers and interest to protect them. If the attack's target or its ISP could accurately detect attacks and devise accurate filtering rules, these rules can be installed automatically, and for free, in existing firewalls and switches, at the target or at their first-hop ISP. The research gap here is that there are no publicly available solutions, which detect DDoS attacks at scale (e.g., at ISP-size networks with millions of potential targets), and produce accurate network signatures for filtering.

We propose AMON-SENSS, an open-source, scalable, flow-level DDoS detection for ISPs. AMON-SENSS employs *binning* to make detection scalable at the ISP level, and employs layers of bins to reduce false alerts. AMON-SENSS works offline using existing flow capture tools, analyzes traffic at multiple granularities and detects an anomaly when both volume and asymmetry of the traffic in a bin increase for a sustained time period. While anomaly-based DDoS detection has been widely explored in the past (e.g., [4]–[6]), the novelty of AMON-SENSS lies in scalability of its detection, and the ability to produce accurate attack signatures, which can be immediately deployed at existing firewalls. This deployability (offline detection using existing traffic captures, ready-to-deploy firewall rules) distinguishes AMON-SENSS from other research approaches, which classify traffic flows instead of producing attack traffic signatures, and are much less deployable. Flow classification must work in line with traffic, so flows classified as attack can be immediately filtered. Inline deployment requires new hardware. Many classification approaches also deploy machine learning, and thus deployment must support line-rate machine learning, which network devices do not support today.

We further show that AMON-SENSS offers superior performance compared to two commercial defense solutions, NetScout [7] and FastNetMon [8]. Using 37 days of real traffic from a mid-size ISP FRGP, AMON-SENSS outperforms commercial approaches both in accuracy (F1-score of 92–99% versus NetScout's 26–70% and FastNetMon's up to 2%) and in detection speed (AMON-SENSS's detection delay is less than a third of the NetScout's). We release AMON-SENSS as open-source at [9].

978-1-6654-3540-6/22 

## II. Related Work

Many solutions to detect DDoS attacks have been proposed in literature [10]. We discuss only those research approaches that are closely related to AMON-SENSS in this section.

**Packet-based solutions:** These solutions learn feature distribution (e.g., packet length, port numbers, etc.) in legitimate traffic, prior to attacks, and use these models to classify each packet as attack or legitimate. PacketScore [11], Carl et al. [4], Fouladi et al. [12], Feinstein et al. [6], Lotfollahi et al. [13], Yuan et al. [14], Bardas et al. [15], and Kitsune [16] are examples of packet-based approaches. These approaches can be very accurate, but they are not practical. First, many networks only collect sampled NetFlow data, and not packet data, due to scale challenges. Second, packet classification approaches must be installed inline, so that they can immediately filter packets they identify as attacks. Machine learning and line-rate packet classification require special hardware, which many networks do not have today.

**Flow-based solutions:** Similar to packet-based solutions, some flow-based solutions focus only on flow classification problem, using clustering or machine learning (e.g., [5]). These approaches have the same deployment challenge as packet-based approaches – they must be deployed inline on special hardware. Other flow-based approaches work to identify *attack sources*, such as Braga et al. [17], Simpson et al. [18], Xu and Liu [19], and Doshi et al. [20]. Source identification is useful, but not practical for mitigation, as attack sources can be spoofed or there can be more sources than filtering rules that can fit in today's switches [21].

None of the related work approaches produces network-level signatures in terms of transport ports and protocols, for attack filtering – they simply identify the attack's onset and in some cases the attack's target and sources, which is not useful for operational DDoS defenses. Our focus is on identifying best, concise network filtering rules via offline traffic processing. Such rules can then be efficiently deployed in standard network hardware, such as switches and firewalls. In [22], Wagner et al. propose collaborative DDoS detection, involving multiple ISPs, where each ISP applies simple, threshold-based detection rules, and exchanges detection signals with others. Our work is complementary to Wagner et al. [22], and could replace their threshold-based detection.

## III. DDoS Attack Detection

DDoS attacks come in many flavors [10], [23]–[25]. *Volumetric attacks*, such as UDP flood attack, are attacks that overwhelm internal network capacity or even centralized DDoS mitigation scrubbing facilities with significantly high volumes of malicious traffic. *Application-level attacks*, such as flash-crowd attacks [26] or exploit-based attacks, are attacks that overload specific application resources with legitimate looking requests to make the application unavailable or unresponsive to legitimate users.

Because there are many DDoS variants, it is difficult to design a solution that detects them all reliably. In this paper *we focus on detection of volumetric attacks*, i.e., those attacks that create a visible increase of volume of traffic to their target. One might consider such attacks trivially detectable, but detection is challenging at the network level. An attack could be too small at the network level to trigger detection, while still being large enough to overwhelm its target. The problem is compounded at large ISPs, which route extremely high traffic volumes in aggregate, and serve millions of customers. *Threshold-based* approaches for attack detection usually employ manually set thresholds per target IP address, and raise alerts when incoming traffic exceeds them. Since different attacks can be harmful at different rates, threshold-based approaches cannot fully address the problem. *Anomaly-based* approaches track legitimate traffic's features for each potential target IP address, and detect deviations as attacks. Such approaches are promising, but fall short in scaling up to millions of traffic streams, and in separating benign traffic spikes from malicious ones.

Once an attack is detected, the defense should produce an accurate network signature of the attack for filtering, specifying IP and transport header fields, e.g., `dst IP 1.2.3.4 and proto udp and src port 53`. While it may be tempting to produce signatures that specify sources of attack traffic, such filtering rules can be bypassed if the attacker uses IP spoofing. Even without IP spoofing, large botnet-launched attacks can lead to thousands of filtering rules—a scale that cannot fit into today's switch TCAM [21]. AMON-SENSS focuses on generation of network-level signatures, which specify attack type and target, but not attack sources.

## IV. AMON-SENSS

In this section, we describe AMON-SENSS. We discuss how we keep scalable statistics in Section IV-A, our features in Section IV-B, and our anomaly detection in Section IV-C. Our proactive signature generation is described in Section IV-D and our alarm generation in Section IV-E. We denote an IP address as *local address* if it belongs to an ISP customer running AMON-SENSS (*foreign address* otherwise).

### A. Binning

AMON-SENSS learns models of legitimate traffic during training, and deploys them continuously after the first training period to perform detection. It also continuously collects data, and updates existing models at customizable time intervals, if no anomaly has been detected. AMON-SENSS must collect and keep some traffic statistics to build models of legitimate traffic. To achieve scalable operation at the ISP level, we adapt the idea of *binning* traffic statistics, proposed by Kallitsis et al. in AMON [27]. AMON bins traffic per source and destination IP address. Bins are organized as a matrix and an IP is hashed to provide the index of the row (source IP) or column (destination IP) in the matrix. AMON periodically emits a list of heavy-activity bins that can be used for traffic engineering purposes, accounting and security forensics. It leverages the Boyer-Moore majority vote algorithm [28] to identify heavy-hitters in each bin, which could be DDoS attack targets or attack sources (e.g., scanners).

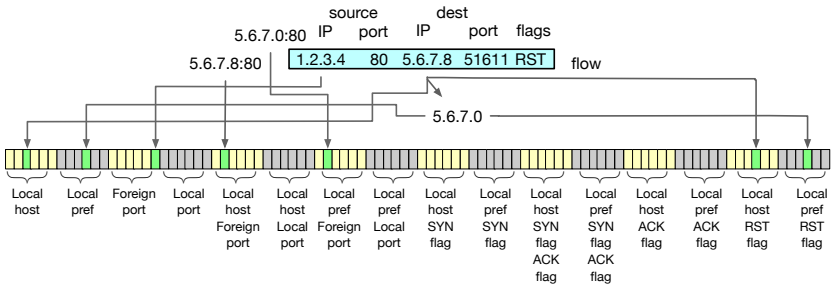
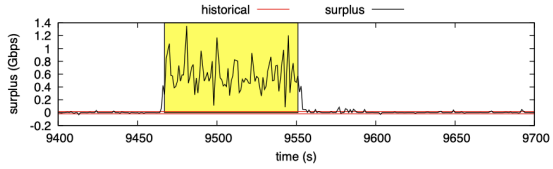



Fig. 1. Binning in AMON-SENSS: yellow and grey segments are different bin arrays, and the green bins will be updated with the given flow's statistics.

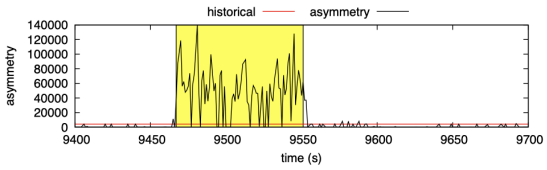
We repurpose AMON's binning idea for DDoS attack detection and signature generation. We do not track behavior of foreign hosts, but only that of local hosts. We design our binning process in a way that helps us devise network signatures for flow filtering, to mitigate DDoS attacks. Since network signatures specify IP addresses, ports, transport protocol and TCP flags, we focus on these fields when binning traffic. Because we want to be able to detect attacks on an IP address or the entire prefix, some of our bins collect statistics per local address, while others collect statistics per local /24 prefix. Some attacks involve a specific service port (e.g., reflection attacks or SYN flood attacks). To detect these attacks, some of our bins track traffic to and from local and foreign service ports. Some attacks also involve sending a large number of TCP packets with specific flags (SYN, SYNACK, RST or ACK). We use dedicated bins to track this dimension as well.

Overall, we create 16 bin arrays, each of the same size, and call the whole structure a *bin layer*. An array tracks one or a combination of traffic flow features, e.g., local IP, local service port, combination of local IP and service port, etc. Let us denote local host as LH, local prefix as LP, local port as Lp, foreign port as Fp, and TCP flags as SYN, SYNACK, ACK and RST. Each bin array tracks traffic aggregated along one of the following flow features: (1) LH, (2) LP, (3) Fp, (4) Lp, (5) LH and Fp, (6) LP and Fp, (7) LH and Lp, (8) LP and Lp, (9) LH and SYN, (10) LP and SYN, (11) LH and SYNACK, (12) LP and SYNACK, (13) LH and ACK, (14) LP and ACK, (15) LH and RST, and (16) LP and RST. For example, bin array 9 tracks SYN flood traffic to local hosts, and bin array 5 tracks traffic to local hosts from a specific foreign port. Each bin in a bin array stores statistics for a set of flow feature values that hash into that bin. Each flow contributes to some of the 16 bin arrays, depending on the flow's network features. We combine flow features of interest for a given bin array into *bin key*, and use a simple and fast modulo function using bin array size, to calculate the bin index from the bin key.

Since multiple flows are binned into the same bin (e.g., bin 18 in the first bin array will hold aggregate statistics for traffic to and from all local hosts whose IP address modulo bin array size equals 18), false positives and false negatives are possible. We address this problem in two ways. First, during training we work to identify heavy hitter bin keys in each bin array and dedicate to them additional, individual bins, instead of grouping them with other keys. Second, we apply ideas




(a) Surplus




(b) Asymmetry

Fig. 2. Illustration of attack detection using statistics from one select bin in our dataset.

from Bloom filters [29] to reduce collisions, and run AMON-SENSS with multiple bin layers, each using a different bin array size. This is similar to using multiple hash functions in Bloom filters.

### B. Features

AMON-SENSS keeps statistics for two features per bin: traffic surplus (in bytes) and traffic asymmetry. All statistics are calculated with regard to local hosts/prefixes. Let $B_r$ and $P_r$ be traffic volume and packets received by a local host/prefix, and $B_s$ and $P_s$ be volume and packets sent by a local host/prefix. Traffic surplus is calculated as: $sur = B_r - B_s$, while traffic asymmetry is calculated as $asym = P_r/P_s$.

Flow surplus indirectly measures the amount of traffic received by local hosts/prefixes. We use surplus, instead of received traffic, to differentiate between flows that engage local servers (negative surplus, since server responses are larger than client requests) and flows that engage foreign servers (positive surplus, when clients become more active). DDoS attacks will create a large traffic surplus. *Traffic surplus increase is not sufficient to detect DDoS attacks*. A local host may receive more traffic, because it became more active, or because it is under attack.

Asymmetry measure captures the ability of the host to handle the traffic it receives. A local server may process many service requests, and thus have large surplus, but low asymmetry. A local client may contact many servers, also leading to large surplus, but low asymmetry. *Increased asymmetry itself is not sufficient to detect DDoS attacks*. A host could receive scans, which would lead to high asymmetry, but no

significant increase in traffic surplus, and should not trigger DDoS mitigation. Only when a local host exhibits a sudden increase in both surplus and asymmetry, can we infer that it receives *unwanted traffic that it cannot handle*, i.e., that it may be under a DDoS attack.

### C. Anomaly Detection

AMON-SENSS detects a possible DDoS attack when both traffic surplus and asymmetry in a bin are anomalous. At first, AMON-SENSS goes through a training period, and learns means and standard deviations of traffic surplus and asymmetry for each bin—we call these *historical profiles*. After training, AMON-SENSS collects traffic measurements for each bin and builds *instantaneous profiles*. After each instantaneous profile update, AMON-SENSS runs the CUSUM algorithm [4] to detect if the current traffic surplus and asymmetry are anomalous, compared to the historical profile. If so, the update is rolled back and we increase an *anomaly score* for the bin. If both traffic surplus and asymmetry are normal, we decrease the anomaly score for the bin. When the anomaly score exceeds a score threshold $S$, AMON-SENSS detects a possible DDoS attack. The score threshold's value is correlated with our detection delay. Larger values lead to higher detection delay, but they reduce false positives and avoid detection of attacks that are too short, and may end before we can trigger mitigation. Our detection approach is illustrated in Figure 2. Red lines represent historical values for surplus and asymmetry for one select bin, and black lines represent the instantaneous values over time. The yellow rectangle highlights the times when both surplus and asymmetry exceed their historical values, signaling an attack.

### D. Signature Generation

As the anomaly score for the bin starts increasing, we *proactively* generate and evaluate potential attack signatures. Signatures are generated using only flows received by local hosts. We generate seven types of *signature candidates* for each flow that is binned, by masking or unmasking three fields of the flow identifier: destination IP address, source port and destination port. The flow's transport protocol is always included in the signature instance, and we do not include source IP addresses. Based on the bin type, some flow identifier fields or other flow features may be included. For example, bins in layer 9 (LH and SYN) lead to signature candidates that all include TCP SYN flag. Each signature candidate collects *score points* for each packet matching this signature in the current monitoring period.

Since multiple flows are binned into a given bin, we would quickly accumulate many signature candidates, which would increase memory cost. We address this by running Boyer Moore majority vote algorithm [28] to only keep one signature candidate per type (seven candidates per bin), with candidate's score points used as votes. The best signature candidate per type thus matches most packets that contributed to the given bin. When an attack is detected by AMON-SENSS in one specific bin, all seven signature candidates from that bin are evaluated to find the best one. We identify the candidate that is the most specific (to minimize collateral damage), and whose score points represent a large fraction (customizable parameter, default value is set to 0.75) of packets received by that bin, to ensure that filtering will be effective.

### E. Visible Alert Generation

An anomalous bin with a non-empty signature leads to alert generation by AMON-SENSS. Alerts are generated in plaintext logs, one log line per bin layer. Each time an alert is generated and written into a log, the anomaly signal is reset for the given bin in a given layer. Ongoing attacks can result in multiple, repeated alerts being logged.

Simultaneously with alert logging, we apply post-processing to generate *visible alerts*, to display to operators. Visible alerts are generated from the AMON-SENSS log by: (1) exporting alerts that appear simultaneously in all bin layers, (2) aggregating alerts, and (3) pruning weak alerts.

**Export.** We identify candidate alerts that appear simultaneously in all bin layers. Such alerts proceed to aggregation and pruning.

**Aggregation.** Since traffic statistics are collected in multiple bin arrays (in one given bin layer), a given attack can create anomalies in several arrays. For example, an NTP flood to a target may lead to alerts from bin arrays that monitor traffic to local IP, local prefix, foreign port and the combinations of local IP/local prefix and foreign port. Further, a target may be hit simultaneously with multiple attacks. During aggregation, we note the target for each alert, and aggregate all alerts pertaining to this target within a given time interval $T_{on}$ into a single *candidate alert*. That *candidate alert* is considered active, and goes into the pruning stage, with the *number of votes* denoting how many alerts were aggregated into it.

**Pruning.** We prune candidates that have fewer votes than a customizable threshold $V$. This helps suppress alerts for short attacks, which do not warrant mitigation. The remaining alerts become visible to operators. In the background, we track each visible alert and deactivate it (generate "attack stopped" message to operators) when it stops receiving new votes for period $T_{\text{off}}$.

## V. Evaluation

We evaluate AMON-SENSS on Netflow traffic traces collected in a mid-size US ISP, FrontRange GigaPOP (FRGP), connecting educational, research, government and business institutions to the Internet. Traces are collected on all ingress/egress links between FRGP and the Internet, using packets sampled at either 1:100 or 1:4096 rate, and cover 37 days in 2020. We provide further details in Table I. In addition to NetFlow traces, we also have alerts generated by a commercial DDoS appliance, NetScout, which is deployed on one, large ingress/egress link between FRGP and its upstream ISP. NetScout's alerts denote start and stop times of the attacks (as observed by NetScout), the alleged target and the attack type, which can be easily transformed into a network-level signature. We are in discussions with FRGP to release

TABLE I
DATASETS USED IN OUR EVALUATION

| month | days | app. flows | app. bytes |
|---|---|---|---|
| May 2020 | 9 | 2 T | 30 PB |
| August 2020 | 15 | 3.8 T | 45 PB |
| September 2020 | 13 | 3.3 T | 40 PB |

anonymized dataset (NetFlow traffic with ground truth labels, and NetScout alerts) we used in evaluation to public. Once available, the dataset will be released at [30]. We evaluate AMON-SENSS against NetScout, and against an open-source NetFlow-based DDoS detection engine, FastNetMon [31], which is the basic, free version of the same-named commercial tool [8]. While details of how NetScout and FastNetMon detect attacks and determine attack type are proprietary, various technical brochures and blog posts indicate some use of threshold-based detection [32].

*1) Ground Truth Labeling:* Since we want to compare NetScout with AMON-SENSS, we cannot use NetScout's alerts as ground truth. Instead we apply the approach described in [22], [33] to identify *reflection attacks* in our datasets as events where at least 10 different sources send more than 100 Mbps of traffic to a given target within the same second, using the same source port. This approach was shown to be accurate in Kopp et al. [33], and was also used in Wagner et al. [22]. While accurate for ground truth labeling, this approach requires tracking separately traffic to each of the millions of ISP customers and thousands of ports per customer. It is memory- and CPU-hungry, running longer than real time and consuming up to 16 GB of memory on our datasets, and it cannot be used for scalable DDoS detection. It is also limited to reflection attacks only.

*2) Evaluation Approach and Calibration:* In our evaluation we compare AMON-SENSS to NetScout and to an open-source version of FastNetMon [31]. We compare the detection delay of these approaches, their accuracy, and their signature quality. We focus only on detections of reflector attacks, which exceed 100 Mbps, because we have ground truth for these attacks. Both AMON-SENSS and FastNetMon use multiple parameters that determine their sensitivity and accuracy. AMON-SENSS uses: anomaly score threshold $S$, number of bins per bin array $N$, number of layers $L$, times for attack alert aggregation $T_{on}$ and $T_{\text{off}}$, and vote threshold for pruning $V$. FastNetMon uses: attack detection threshold in terms of packets per second $threshold\_pps$, attack detection threshold in terms of volume $threshold\_mbps$, and duration to keep an attack source in blocked state $ban\_time$. We calibrate these parameters on three days of data in the September dataset, and select best-performing settings for full evaluation. Our settings for AMON-SENSS are: $S$=60, $N$=3337, $V$=10, $L$=5, $T_{on}$=20 s, $T_{\text{off}}$=60 s. Our settings for FastNetMon are: $ban\_details\_records\_count$=500, $ban\_time$=3600s, $threshold\_pps$=20,000.

*3) Results:* Table II shows our results for AMON-SENSS, NetScout and FastNetMon. AMON-SENSS achieves superior performance, with F1-score of 92%–99%, and low or no false positives and false negatives. NetScout has a significantly

TABLE II
RESULTS: A-S: AMON-SENSS, NS - NETSCOUT, FNM - FASTNETMON

| data | detect. | gr. truth | TP | FP | FN | F1 | delay |
|---|---|---|---|---|---|---|---|
| **May** | A-S | 12 | 12 | 2 | 0 | 0.92 | 27 s |
| | NS | | 7 | 1 | 5 | 0.70 | 99 s |
| | FNM | | 11 | 26.5 K | 1 | 0.0008 | 43 s |
| **Aug** | A-S | 39 | 38 | 5 | 1 | 0.92 | 17 s |
| | NS | | 7 | 8 | 31 | 0.26 | 75 s |
| | FNM | | 24 | 13.5 K | 15 | 0.004 | 61 s |
| **Sep** | A-S | 126 | 126 | 3 | 0 | 0.99 | 20 s |
| | NS | | 38 | 15 | 70 | 0.47 | 66 s |
| | FNM | | 106 | 11.6 K | 20 | 0.02 | 78 s |

lower F1-score of 26–70%, mostly due to large false negatives. NetScout's false negatives occur either on short attacks (under one minute) or on attacks that are under 1 Gbps, potentially due to thresholding issues. FastNetMon fares the worst—it has many false positives, leading to F1-score of 2% or less. FastNetMon fails to identify multiple reflection attacks, such as CLDAP amplification. With regard to detection delay, AMON-SENSS has a detection delay of under 30 seconds from attack onset, while NetScout and FastNetMon take 2–3 times as long. We further evaluate the quality of signatures for AMON-SENSS, NetScout and FastNetMon by comparing them with ground-truth signatures for all true positives. When a ground truth attack involves several attack vectors, a signature may match the ground-truth partially (only for some vectors) or fully (for all vectors). AMON-SENSS achieves 60% full matches and 40% partial matches, compared to NetScout's 57% full and 43% partial matches, and FastNetMon's 100% partial matches. AMON-SENSS thus clearly produces the most accurate signatures. Another way to measure signature quality is to evaluate how well the signature filters attack traffic. We measure the amount of attack traffic dropped by AMON-SENSS, NetScout or FastNetMon, and compare it to the ideal case, which uses signatures derived from the ground truth. AMON-SENSS filters 80% of the ground-truth attack traffic, while NetScout filters only 46%, due to larger detection delay and some false negatives. FastNetMon filters 56% of the ground-truth attack traffic. AMON-SENSS further achieves 98% precision (98% of filtered traffic is indeed attack, 2% is dropped due to false positives), compared to NetScout's 99%, and FastNetMon's 71%. Thus overall, AMON-SENSS's signatures have the highest quality.

*4) Sensitivity Analysis:* In this Section we explore how performance of AMON-SENSS depends on values of its parameters, using select three days from the September dataset. For space reasons, we summarize results.

**Score threshold $S$.** We experimented with thresholds for anomaly score 5–60. Lower values slightly increased false positives.

**Number of bins per bin array $N$.** We explored values 1 K–64 K. Larger values reduce errors (false positives and false negatives), but the gain is small. False negatives stabilize once we exceed 3 K bins. False positives are better handled by increasing the number of layers than by increasing the number of bins.

**Number of bin layers $L$ and voting threshold $V$.** We jointly explored these parameters, evaluating 1–10 bin layers

and $V$={1-20}. Increasing the number of layers and voting threshold improves F1-score, but the improvement decreases above 5 layers, and $V$=10.

**Alert aggregation times $T_{on}$ and $T_{\text{off}}$.** We explore values 5–200 sec. for these two parameters. While $T_{\text{off}}$ variation does not change AMON-SENSS's performance, increasing $T_{on}$ increases false positives, while lowering detection delay. Optimal values for $T_{on}$ are 20–30 s.

*5) Operational Cost:* On Intel Xeon 3.2GHz CPU with 4 cores, AMON-SENSS processes a day of traffic in seven hours, with 10 layers and 3,337 bins per bin array, consuming up to 11 GB of memory. Speed and memory cost scale linearly with $N$ and $L$ parameters. For example, with 1 layer and 3,337 bins, it takes around 0.75 hours to process a day of traffic. Thanks to binning, memory cost stays constant when the size of the monitored network or traffic volume change.

## VI. Conclusions

In this work, we propose a scalable, accurate, open-source DDoS detection system, AMON-SENSS. AMON-SENSS employs binning and layering to collect traffic statistics in a scalable manner, and observes traffic at multiple granularities. AMON-SENSS applies anomaly detection using traffic surplus and asymmetry, and proactively collects and evaluates network level signatures. We evaluate AMON-SENSS against two commercial defense systems using 37 days of real traffic from a mid-size ISP. We find that AMON-SENSS exhibits superior performance in terms of accuracy, latency and network signature quality over these commercial alternatives, and we release it as open-source at [9].

## VII. Acknowledgment

This material is based on research sponsored by the Department of Homeland Security (DJ-IS) 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 UK of Great Britain and Northern Ireland via contract number D15PC00184. 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 UK of Great Britain and Northern Ireland.

## References

[1] Imperva, 2020, https://tinyurl.com/y5jmjuzv.
[2] "DDoS Attack Trends for Q4 2021." https://tinyurl.com/tkjy584w.
[3] "The Long Tail of Attacker Innovation." https://tinyurl.com/yp74j3f8.
[4] G. Carl, G. Kesidis, R. Brooks, and S. Rai, "Denial-of-service attack-detection techniques," *IEEE Internet Computing*, vol. 10, no. 1, pp. 82–89, 2006.
[5] X. Qin, T. Xu, and C. Wang, "DDoS Attack Detection Using Flow Entropy and Clustering Technique," in *International Conference on Computational Intelligence and Security (CIS)*, 2015, pp. 412–415.
[6] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to DDoS attack detection and response," in *Proceedings DARPA Information Survivability Conference and Exposition*, vol. 1, 2003, pp. 303–314 vol.1.
[7] "Arbor DDoS," https://www.netscout.com/arbor-ddos.
[8] "FastNetMon," https://fastnetmon.com/.
[9] "AMON-SENSS code," https://github.com/jelenamirkovic/AMON-SENSS.
[10] T. Mahjabin, Y. Xiao, G. Sun, and W. Jiang, "A survey of distributed denial-of-service attack, prevention, and mitigation techniques," *International Journal of Distributed Sensor Networks*, vol. 13, no. 12, p. 1550147717741463, 2017.
[11] Y. Kim, W. C. Lau, M. C. Chuah, and H. Chao, "Packetscore: a statistics-based packet filtering scheme against distributed denial-of-service attacks," *IEEE Transactions on Dependable and Secure Computing*, vol. 3, no. 2, pp. 141–155, 2006.
[12] F. Fouladi, E. Kayatas, and E. Anarim, "Statistical measures: Promising features for time series based DDoS attack detection," in *MDPI Proceedings*, vol. 2, no. 2, 2018, p. 96.
[13] M. Lotfollahi, J. Siavoshani, H. Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, no. 3, pp. 1999–2012, 2020.
[14] X. Yuan, C. Li, and X. Li, "DeepDefense: identifying DDoS attack via deep learning," in *SMARTCOMP*. IEEE, 2017, pp. 1–8.
[15] A. G. Bardas, L. Zomlot, S. C. Sundaramurthy, X. Ou, S. R. Rajagopalan, and M. R. Eisenbarth, "Classification of UDP Traffic for DDoS Detection." in *LEET*, 2012.
[16] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," *NDSS*, 2018.
[17] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *IEEE Local Computer Network Conference*, 2010, pp. 408–415.
[18] K. A. Simpson, S. Rogers, and D. P. Pezaros, "Per-host DDoS mitigation by direct-control reinforcement learning," *Transactions on Network and Service Management*, vol. 17, no. 1, pp. 103–117, 2019.
[19] Y. Xu and Y. Liu, "DDoS attack detection under SDN context," in *IEEE INFOCOM*. IEEE, 2016, pp. 1–9.
[20] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," in *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018, pp. 29–35.
[21] R. Li, Y. Pang, J. Zhao, and X. Wang, "A tale of two (flow) tables: Demystifying rule caching in openflow switches," in *International Conference on Parallel Processing*, ser. ICPP 2019, 2019.
[22] D. Wagner, D. Kopp, M. Wichtlhuber, C. Dietzel, O. Hohlfeld, G. Smaragdakis, and A. Feldmann, "United we stand: Collaborative detection and mitigation of amplification ddos attacks at scale," in *ACM SIGSAC CCS*, 2021, pp. 970–987.
[23] J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.
[24] A. Srivastava, B. Gupta, A. Tyagi, A. Sharma, and A. Mishra, "A recent survey on ddos attacks and defense mechanisms," in *International Conference on Parallel Distributed Computing Technologies and Applications*. Springer, 2011, pp. 570–580.
[25] R. Tandon, "A survey of distributed denial of service attacks and defenses," 2020, https://arxiv.org/pdf/2008.01345.pdf.
[26] R. Tandon, A. Palia, J. Ramani, B. Paulsen, G. Bartlett, and J. Mirkovic, "Defending web servers against flash crowd attacks," in *ACNS*. Springer, 2021, pp. 338–361.
[27] M. Kallitsis, S. Stoev, S. Bhattacharya, and G. Michailidis, "AMON: An open source architecture for online monitoring, statistical analysis, and forensics of multi-gigabit streams," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 6, pp. 1834–1848, 2016.
[28] R. S. Boyer and J. S. Moore, "MJRTY—a fast majority vote algorithm," in *Automated Reasoning*. Springer, 1991, pp. 105–117.
[29] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet mathematics*, vol. 1, no. 4, pp. 485–509, 2004.
[30] "COMUNDA – Community Understanding of Network Datasets," https://comunda.isi.edu.
[31] "FastNetMon," https://github.com/pavel-odintsov/fastnetmon.
[32] "Network Telemetry for DDoS," https://linkmeup.ru/blog/927/.
[33] D. Kopp, C. Dietzel, and O. Hohlfeld, "DDoS never dies? An IXP perspective on DDoS amplification attacks," in *International Conference on Passive and Active Network Measurement*. Springer, 2021, pp. 284–301.