Computer Networks 121 (2017) 25–36

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Detecting HTTP-based application layer DoS attacks on web servers in the presence of sampling

Hossein Hadian Jazi, Hugo Gonzalez, Natalia Stakhanova*, Ali A. Ghorbani

Faculty of Computer Science, University of New Brunswick, Canada

ARTICLE INFO

Article history:
Received 17 October 2016
Revised 10 February 2017
Accepted 24 March 2017
Available online 30 March 2017

Keywords:
Denial-of-service attacks
Application layer
Sampling techniques
Intrusion detection
DDoS
Network security

ABSTRACT

A recent escalation of application layer Denial of Service (DoS) attacks on the Internet has quickly shifted the interest of the research community traditionally focused on network-based DoS attacks. A number of studies came forward showing the potency of attacks, introducing new varieties and discussing potential detection strategies. The underlying problem that triggered all this research is the stealthiness of application layer DoS attacks. Since they usually do not manifest themselves at the network level, these types of attacks commonly avoid traditional network-layer based detection mechanisms.

In this work we turn our attention to this problem and present a novel detection approach for application layer DoS attacks based on nonparametric CUSUM algorithm. We explore the effectiveness of our detection on various types of these attacks in the context of modern web servers. Since in production environments detection is commonly performed on a sampled subset of network traffic, we also study the impact of sampling techniques on detection of application layer DoS attack. Our results demonstrate that the majority of sampling techniques developed specifically for intrusion detection domain introduce significant distortion in the traffic that minimizes a detection algorithm's ability to capture the traces of these stealthy attacks.

1. Introduction

The frequency and power of Denial-of-Service (DoS) attacks continue to break records. 2015 was marked as the worst year for DoS attacks in history with attacks reaching 500 Gbps [4]. The year of 2016 however broke that record with reported attacks of 800 Gbps [4]. Leveraging botnets and high-speed network technologies, modern DoS attacks exceed the scale of 400 Gbps becoming a major threat on the Internet [39]. Being one of the oldest type of attacks on the Internet, DoS attacks are known for their disruptiveness and ability to deplete the computing resources and/or bandwidth of their victims in a matter of minutes. In spite of being trivial in execution, they are easily detectable mostly due to their dynamic and voluminous attack rates. As a result in recent years the security community has seen an unprecedented increase in application layer attacks. One of these is a rapidly growing category of application layer Denial-of-Service (DoS) attacks, representing 16% of all attacks in 2016 [4].

As oppose to traditional DoS attacks, application layer DoS attacks are perceived as stealthy, sophisticated and practically undetectable at the network layer [6,51]. Focusing on specific characteristics and vulnerabilities of application layer protocols, application layer DoS attacks are capable of inflicting the same level of impact as traditional flooding DoS attacks at a much lower cost.

With the latest escalation of application-layer DoS attacks, the research community has focused its attention on defense and mitigation techniques for these type of attacks. As opposed to network layer DoS attacks, detection of application layer attacks is a less studied area. The majority of the proposed techniques are only applicable to particular application layer DoS attacks types (e.g., detection of XMLDoS [13], SIP DoS attacks [40]) or flooding attacks. None of these methods offer an inclusive approach for detection of various types of application layer attacks.

In this work we explore several types of application-layer DoS attacks and propose a generic detection approach based on nonparametric CUSUM algorithm. This algorithm was previously applied for detection of voluminous DDoS attacks at the network layer by Chen et al. [11]. We explore the effectiveness of detection approach in a broader scope and on various types of attacks in the context of modern web servers as they represent the most common target for application DoS attacks. Designing proper defense mechanisms for detection of these attacks requires an un-

* Corresponding author.
E-mail addresses: h.haidan@unb.ca (H.H. Jazi), hugo.gonzalez@unb.ca (H. Gonzalez), natalia.stakhanova@unb.ca (N. Stakhanova), ghorbani@unb.ca (A.A. Ghorbani).

http://dx.doi.org/10.1016/j.comnet.2017.03.018
1389-1286/

derstanding of attack attributes. As such we look at characteristics commonly seen as defining for this class of attacks such as limited resource consumption on an attacker's side, targeted damage, and attack stealthiness. To assess the effectiveness of the proposed detection strategy we experiment with two data sets (a prepared traffic with a mix of application layer DoS attacks and a real traffic collected from a web server residing on an academic network). Our experiments show that the proposed scheme is precise and has high accuracy detecting even stealthy attacks.

Increasing complexity of network infrastructure aggravated by constantly growing capacity of networks presents a number of challenges to traditional intrusion prevention and detection devices that are struggling to cope with the sheer amount of data. This difficulty is generally resolved with the use of sampling that allows to address the scalability concerns and consequently, apply traditional intrusion detection approaches. Since application of sampling is practically universal in the modern network infrastructures (e.g., Cisco, Juniper, and Fortinet employ random n-out-of-N sampling as a default technique), it is important to understand the impact of sampling techniques on the proposed detection method of application-layer DoS attacks. In this context we investigate thirteen sampling techniques: eight methods designed for broad traffic estimation and five techniques specifically developed for anomaly detection.

Our findings show that selective flow sampling and sketch-guided sampling methods are the most effective techniques capable of retaining sufficient traces of application layer attacks. Although computationally intensive, selective flow sampling shows the best performance outperforming all analyzed techniques with 100% accuracy and no false positive alerts for sampling rates above 20%. For smaller sampling rates, a generic sketch-guided sampling presents itself as an ideal candidate for the detection of application layer attacks. Our results also show that the specialized sampling techniques designed for anomaly detection context significantly distort traffic patterns limiting detection algorithm's ability to accurately identify these attacks.

To summarize, there are several novel contributions that this study outlines.

- We introduce a generic approach based on a selected combination of application and network level attributes for detection of various types of application layer DoS attacks. We test the proposed method in a realistic setting that allows us to judge approach's capabilities.
- We further assess the performance of the proposed detection in the presence of sampling and investigate impact of sampling methods on detection of application level DoS attacks. It is the first attempt to provide analysis of such a diverse set of sampling methods for intrusion detection domain.

The remainder of the paper is organized as follows. Section 2 presents related work covering both existing literature in detection of application layer DoS attacks and sampling methods. Section 3 gives a brief explanation of characteristics of application layer attacks and Section 4 presents our approach to their detection. In Section 5 we briefly introduce selected sampling methods. Section 6 gives an overview of employed datasets and results of detection, followed by the detailed evaluation of sampling impact in Section 7. Section 8 concludes the paper.

2. Related work

The recent escalation of application-layer DoS attacks have attracted a significant interest of a research community. Since application-layer attacks usually do not manifest themselves at the network level, they avoid traditional network-based detection mechanisms. As such, security community focused on specialized application-layer DoS attacks detection mechanisms. These research efforts can be broadly divided into several groups: *application-based*, *puzzle-based approaches* and *network traffic characteristics based*.

Application-based techniques are generally geared toward legitimate and thus expected characteristics of an application behavior. These approaches include detection of deviations from normal behavior of users browsing web pages [50,53,54], monitoring characteristics of HTTP sessions [41,42], monitoring a number of clients requests [52], and analyzing popularity of certain websites [51]. In many of these approaches, rate-limiting serves as a primary defence mechanism.

Puzzle-based methods are similar to these approaches. However, instead of monitoring characteristics of particular applications, puzzle-based methods, as the name suggests, offer a puzzle to solve and detect potential DoS attack by the ability of the client at the IP addresses to solve it or by their reaction to the offered puzzle. One of these techniques is the detection of attacks using CAPTCHA puzzle [33]. Although this technique may offer a simple approach to attack detection and mitigation, a number of studies showed its ineffectiveness [5,35].

Monitoring characteristics of network traffic for application-layer DoS detection has been suggested by [23] and has been employed for differentiation of flash crowd and true DoS attacks. The approach has also found its application in several studies in a form of IP address monitoring [7,36].

Most of these studies deal with general type application-layer denial-of-service attacks. With the introduction of low-rate application-layer DoS attack [27], a number of research efforts were focused on various detection and mitigation techniques [26–29]. Most of these techniques focus specifically on characteristics of incoming network traffic aiming to reveal/prevent patterns specific to low-rate DoS attacks. Macia-Fernandez et al proposed to modify the implementation of application servers in terms of their processing of incoming requests [29].

Application layer DoS attacks also gained attention in industry resulting in a few protection measures. These measures also known as web application firewalls (WAFs) are typically implemented as web server plugins. Focusing on a web traffic at the application layer, WAFs are primarily designed to analyze web application logic. In spite of their sole focus on intrusive web application behavior, only a few of WAF solutions provide inclusive protection against various types of application layer DoS attacks. For example, a popular open source WAF ModSecurity [34] is only capable of detecting attacks triggered with SlowLoris tool. We on the other hand offer a generic technique for detection of various types of application layer DoS attacks at the network layer without accessing host.

The major difficulty of experimentation with application-layer DoS attacks is the lack of datasets, hence in all studies involving low-rate attacks, the authors resorted to either simulation using ns2 and Matlab or evaluation in a strictly controlled environment (e.g., using Minihttpd server or self-implemented servers emulating behavior of an Apache server). As opposed to this approach, we validate the detection effectiveness in a real network environment. We further study the impact of sampling specifically in the context of these attacks.

Over the years a number of sampling techniques were proposed. Most of them were designed for network traffic monitoring and analysis. The early sampling methods are often referred to as static mostly due to their lack of flexibility due to their reliance on a predefined set of rules. One of the first studies in this area was performed by Claffy et al. [15] who proposed three static sampling techniques: *systematic sampling* (taking a first packet in each set), *stratified random* (taking one random packet in each set), and *simple random* (taking n packets from the stream at random).

To mitigate some of the limitations of the static sampling methods (e.g., inability to fully reflect dynamic network behavior), various adaptive techniques were proposed. Choi et al. [12] proposed *adaptive random sampling* technique tailored to the fluctuating nature of traffic, which allows to adjust sampling to a current traffic rate to ensure a minimum error bound. *Sample-and-hold* and *multistage filter* sampling algorithms were proposed to account for 'heavy-hitters', i.e., very large flows that are often underrepresented in sampled traffic [19]. An alternative approach to this problem was presented in *sketch-guided sampling* that suggested to sample packets with a probability decreasing with the size of flow [25]. While these adaptive techniques were primarily targeting packet level sampling, several flow sampling techniques were also introduced. *Smart sampling* (also referred to as threshold sampling [18]) allows for the control the amount of traffic sampled by tuning the probability of a flow to be selected based on its size [17].

A comparative study of packet and flow sampling was conducted by Hohn and Vietch [21]. The study outlines the weaknesses of both approaches suggesting the most suitable application scenarios for both techniques. In general, packet sampling is a simple implementation of a sampling technique requiring less system overhead (i.e, CPU power, memory requirements) than flow sampling which comes at high computational cost. Due to its simplicity, packet sampling is unable to accurately reflect flow statistics and is thus mostly preferred for the collection of basic statistics, while flow sampling allows for the recovery of detailed traffic information.

In spite of this variability, none of the above mentioned approaches were designed for intrusion detection domain. Since anomalies distort the traffic characteristics, sampling algorithms should be robust enough to react to these anomalous changes and provide unbiased estimation of network behavior. Several specialized sampling approaches for anomaly detection were developed. Among them are *Progressive security-aware packet sampling (PSAS)* algorithm that enables a higher sampling rate for malicious packets by scoring traffic maliciousness as it travels through each hop of a network [1]. By design this sampling approach allows to ensure higher detection accuracy, it however requires a deployment of scoring mechanisms throughout a network. Among other algorithms are *Fast filtered sampling* [32], *Adaptive Traffic Sampling Method* [20], *Selective flow-based sampling* [3], and *Adaptive weighted sampling algorithm based on the least squares* [38], which were specifically developed for detection of network-level DoS attacks.

The inability of the sampling algorithms to retain manifestations of attacks has been illustrated in a number of previous studies [2,30,37]. Mai et al. [30,31] explored the performance of four sampling algorithms: random packet and flow sampling, smart and sample-and-hold sampling in the presence of volume and portscan anomalies. While these studies concluded that random flow sampling gives the best accuracy for detection of volume attacks, the authors emphasized the lossy nature of packet sampling techniques. These studies were followed by similar work (using systematic, random n-out-N and uniform random packet sampling) [2] that showed systematic sampling being the worst method for anomaly detection domain. The following study by Brauckhoff et al. [8] showed that packet sampling, although not affecting volumetric anomaly detection much, has an adverse effect on accurate flow estimation.

Although all these studies reported shortcomings of sampling techniques in the context of intrusion detection, none of them studied the effects of sampling methods specifically designed for this purpose. In this work we also address this gap and focus on the impact of both traditional and specialized sampling techniques on the detection of application layer DoS attacks.

3. Application layer DoS attacks

Although cases of application layer DoS attacks have been reported in the past decade (e.g., DNS amplification attack, XML bomb), their recent growth in numbers and variability attracted significant attention of the security community. These attacks target the application-layer protocols and services with almost no affect on network resources. There are several unique features that characterize these attacks [51]:

- *limited resources consumption on an attacker's side*, due to limited capacity at the targeted applications;
- *targeted damage*, due to strategic selection of a target;
- *stealthiness*, due to the intelligent execution of the attack and a necessity to leverage legitimate connections.

Application layer DoS attacks are generally seen in high-volume or low-volume variations.

High-volume attacks. Often referred to as flooding, these attacks are similar in nature to traditional DoS attacks. They are characterized by a high volume of application-layer requests (e.g., HTTP GETs, DNS queries, SIP INVITEs) transmitted to a victim. With rapidly increasing capacities of modern network infrastructures, traditional network level DoS becomes less effective and more resource demanding. On the application layer however, the server's resources become a primary bottleneck making DoS attacks much less resource intensive yet requiring smaller volumes of malicious traffic.

We experiment with high-volume HTTP attacks, generated using HULK (HTTP Unbearable Load King) [22], Ddossim [16] and HTTP flooder goldeneye [45]. All tools generate a bulk of HTTP requests. As opposed to HTTP flooder that offers a simplistic implementation of flood (i.e., HTTP POST flood in our case), HULK aggravates the requests with additional attack vectors randomizing the payloads of various headers, initiating various types of requests, etc. Although Ddossim [16] implements a distributed attack, in essence it is a simple flooding attack with a number of identical requests.

Low-volume DoS attacks. are characterized by small amounts of attack traffic transmitted strategically to a victim. There are three variations of low-volume DoS attacks: *low-rate* attacks, that send traffic in periodic short-time pulses, *slow-rate* attacks, that exploit timing parameters on a server's side by sending/receiving traffic slower than expected, and *one-shot attacks* that inflict damage to a victim with a single connection/request aiming to consume excess amounts of the victim's resources (e.g., Apache Range Header attack [49]). Since one-shot attacks generally exploit a specific weakness or vulnerability in application level protocol/service, in this study we focus our attention on more universal type of application DoS - slow-rate attacks that are often seen in two variations: slow send and slow read.

- **Slow send** is an attack that aims to tie up server resources by slowly sending legitimate incomplete HTTP requests causing a victim server to reserve resources for open connections waiting for their completion. There are several known implementations of this attack with the focus on header or body of request (sometimes refereed to as slow headers or slow body attacks). *Slow headers* attack can be executed with (1) a popular variation implemented in a Slowloris tool [44] that offers partial HTTP requests (with GET or POST methods only) sent at regular intervals to maintain the connection; and with (2) a more general implementation allowing the use of various methods that can be found in Slowhttptest [48]. *Slow body* attack starts with a legitimate HTTP message header and then

continues by sending a HTTP GET or POST payload at a slow pace (e.g., 1byte/ 1 min) [10]. The slow body attack can be executed with RUDY [43] that requires a presence of forms on a web page or with Slowhttptest [48] that implements generic version without a need of web forms.

- **Slow read** variation of an attack starts with a legitimate HTTP request from an attacker to a victim server followed by a slow consumption of the HTTP response sent by a victim. The successes of the attack lies in the ability to maintain a number of active concurrent connections with a server. The type of control over the duration of a connection is operating system dependent and often includes varying the reading rate and minimizing the buffer size on the receiver side. This attack is implemented in Slowhttptest [48] tool.

In this study we primarily experiment with HTTP-based application layer attacks as they constitute the most common type (86% [39]) of application layer DoS attacks.

4. Detection of application layer DoS attacks

Application level DoS attacks are in essence deviations from normal behavioral patterns, i.e., anomalies. In this light we face two challenges: computationally intensive nature of anomaly detection algorithms and stealthiness of application layer attacks.

Although it has been repeatedly emphasized that application layer DoS attacks do not manifest themselves at the network layer [51], we analyzed various application and network level attributes that can potentially retain traces of attacks. Close analysis of flows produced by various types of DoS attacks at the application layer revealed shifts in values of application features (e.g., number of application layer requests). This indication motivated us to investigate , i.e, methods that aim to detect abrupt changes in the observed stream of sequential values.

A nonparametric cumulative sum (CUSUM) procedure commonly used for detection of wide range of possible shifts and is generally favored for its simplicity and low computational overhead [9]. In addition to this, CUSUM-based detection was previously proven to be effective for detection of flooding DoS attacks at the network level [11]. In this work, we investigate a broader detection scope and employ CUSUM-based detection to different types of DoS attacks (including flooding attacks) at application layer.

The idea of the algorithm is to detect changes in distribution of observed values as quickly as possible. Consider a legitimate HTTP session, a sequence of HTTP requests received by a target is typically balanced by a number of corresponding network packets carrying these requests. Since the slow-send attack needs to prolong the connection without actually sending any useful information, a launch of attack would be manifested with a sudden change in the number of requests. This abrupt change in observed number of requests is suspicious, but not necessarily malicious and is thus can be only detected if monitored in combination with other conditions, e.g., the number of corresponding network packets. Intuitively, the CUSUM procedure provides the quick and simple method for detection of these changes.

The standard mathematical formulation of the problem is given as follows (summary of symbols is provided in Table 1): let a sequence of collected observations (e.g., number of packets, requests, sessions) be denoted as $R = \{r_1, r_2, \ldots, r_t\}$. To eliminate sequence dependence on time, R is normalized by an average number of observations Av_t during this time period t, that is $\hat{R}_t = r_t / Av_t$. A recursive estimation of Av_t in real time is given as follows

$$Av_t = \alpha Av_{t-1} + (1-\alpha) r_t \tag{1}$$

The attack occurrence is detected through the sudden changes in $\hat{R}$ that are monitored using a parameter $\beta = \max\{c_i\}, 1 < i < t$,

Table 1
Summary of symbols.

Symbol	Explanation
Av_t	average number of observations during this time period t
$R = \{r_1, r_2, \ldots, r_t\}$	a sequence of collected observations
$\hat{R}_t$	normalized observations at time period t
c_t	the mean over $\hat{R}$
β	parameter estimated as $max\{c_i\}$, $1 < i < t$
S_t	amount of change in a sequence
N	threshold parameter, if $S_t > N$, an attack is flagged
Cwd_t	the number of packets without data
Req_t	the number of requests over given period of time

where c_t is the mean over $\hat{R}$, that can be also estimated recursively as

$$c_t = \alpha c_{t-1} + (1-\alpha)\hat{R}_t \tag{2}$$

Whenever a new value r_t is received, the procedure computes S_t

$$S_t = (S_{t-1} + \hat{R} - \beta)^+ \tag{3}$$

Essentially, $\hat{R} - \beta$ allows to maintain a negative value during normal behavior and become positive when a change is detected. That is if $S_t > 0$, S_t is equal to its current value and 0 otherwise. Intuitively, the larger S_t, the more likely it is that the attack is happening. The extend of this malicious increase is estimated using a threshold parameter N, that is if $S_t > N$, an attack is flagged.

Features. We analyzed various application and network level attributes that can potentially retain traces of attacks. Among them we considered the total number, and also a number of unique packets containing HTTP requests/responses received/send by a server over a specific time interval, number of incoming/outgoing connections/sessions/packets, packets with payloads of various size over a period of time. These features were selected based on the characteristics of application layer DoS attacks, that dictate the necessity to use legitimate connections and manipulate timing parameters of the communication process to remain stealthy.

Throughout our experiments with these features, it became obvious that detecting flooding attacks was straightforward with any of the features, although low-volume slow-rate attacks were more stealthy. The nature of these attacks requires to prolong the connection without encumbering an attacker, which in most cases meant extending the connection with empty packets containing control information in the headers (e.g, change of the buffer size). As such these attacks were exhibiting a specific behavioral patterns that were characterized by disproportionately high number of packets without payload in comparison to number of packets containing application level requests.

As such our detection algorithm was applied on a stream of observed incoming network traffic focusing on two features[1]: the number of application layer requests and the number of packets with payload size equal to zero. To capture the variability of application layer DoS attacks, $\hat{R}$ was calculated as follows:

$$\hat{R} = \frac{Cwd_t}{Req_t + 1} \times Cwd_t \tag{4}$$

where Cwd_t represents the number of packets without data, and Req_t represents the number of requests over given period of time.

5. Sampling techniques

In this study, we focus on the thirteen sampling techniques that include traditional as well as specialized algorithms designed

[1] Due to the space limitations, we only show the experimental results with the best feature set.

Table 2
Overview of sampling techniques.

Sampling technique	Sampling level	Tailored for security domain	Flow/Packet size preference	Notes
Systematic packet sampling [15]	Packet	No	No pref	Implemented in Cisco 12000 series routers [14]
Random packet sampling [15]	Packet	No	No pref	
Random n-out-of-N sampling [15]	Packet	No	No pref	Implemented in Cisco Catalyst switches and 12000 series routers [14], Fortinet FortiGate firewall [46], Juniper JSeries routers [24]
Adaptive random sampling [12]	Packet	No	Medium size	
Random flow sampling	Flow	No	No pref	
Smart sampling [17]	Flow	No	Large	
Sample-and-hold [19]	Hybrid	No	Large	Pkt sampling decision is based on indiv pkt. Amount of sampling is capped on flow basis
Sketch-guided sampling [25]	Hybrid	No	Small/ Medium	Pkt sampling decision is based on the entire flow
Selective flow sampling [3]	Flow	Yes	Small	
Fast filtered sampling [32]	Hybrid	Yes	Small	Pkt sampling decision is based on the entire flow
IP flow-based sampling [37]	Hybrid	Yes	Small/ Medium	Pkt sampling decision is based on the entire flow
Adaptive weighted sampling [38]	Packet	Yes	No pref	
Adaptive traffic sampling [20]	Hybrid	Yes	Small	Pkt sampling decision is based on the entire flow

specially for anomaly detection. A summary of these algorithms is given in Table 2.

Systematic packet sampling is one of the basic traffic sampling techniques [15]. The method works with a count-based sampling interval selecting kth network packet in each interval. Although time-based sampling interval can be also applied, we experimented with count-driven sampling only.

Random n-out-of-N sampling (Stratified random) is another basic static sampling algorithm [15]. Similarly to systematic sampling, it works with sampling intervals of N packets. However, instead of picking the first packet, the algorithm randomly samples one packet from each interval.

Random packet sampling (Simple random) is a static sampling algorithm that randomly selects a network packet in accordance to a predefined probability [15]. Although the selection probability does not necessarily have to be the same for each packet, we experimented with uniform sampling as the most common form of this algorithm.

Adaptive random sampling samples packets with a probability adjusted to a current traffic load dynamics [12]. The algorithm employs AR (Auto-Regressive) model to predict traffic characteristics (i.e., packet sizes, packet counts) of a time block based on measurements from the past blocks. These measurements allow to estimate the packet sampling probability p for the next time block as follows: $p = \frac{\hat{n}}{m}$, where m is the number of packets seen in the current block, and $\hat{n}$ is an estimation of packet sizes sampled in previous blocks using AR model.

Random flow sampling is similar to random packet sampling. Network packets are aggregated in flows based on 5-tuple: ports, IP source and destination addresses, and a protocol (i.e, Netflow format). The resulting flows are sampled with a set probability p.

Smart sampling is a size-dependent flow selection algorithm with the focus on the selection of large flows, i.e, 'heavy-hitters'[17]. The algorithm sampling is guided by a sampling threshold z that indicates the flow size considered to be large. Although there are many ways to select this threshold, the practical approach is to base it on a percentage of available throughput. Formally, a probability of flow Pr of size x to be sampled is calculated as follows:

$$Pr(x) = \begin{cases} \frac{x}{z}, & \text{if } x \le z \\ 1, & \text{if } x > z \end{cases}$$

Sample-and-hold sampling algorithm is similar to random sampling method, with the lean towards larger flows [19]. A packet of size s is sampled with a probability $p_s = 1 - (1-p)^s$. For flows from which packets are sampled an entry in a counting table is maintained. The entries in this table are removed at the end of the sampling interval if the corresponding counter is less than a pre-defined threshold T.

Sketch-guided sampling aims to improve sampling accuracy of small and medium flows [25]. The algorithm proceeds by estimating a size of a flow to which an arrived packet belongs. A packet is sampled with a probability $P(\hat{s})$, where $\hat{s}$ is an estimated flow size. The flow size approximation is performed with the help of so called counting sketches, i.e., a simple data structure that maintains counts of packets per flow. Since the sketches are implemented with the use of hashing, collisions are imminent, which hence leads to approximate flow size.

IP flow-based sampling algorithm [37], although designed specifically for anomaly detection, follows a procedure similar to sketch-guided sampling. A packet is sampled with a probability $p(m) = 1/(1+\epsilon^2 m)$, where m is a position of a packet in arriving sequence for a given flow. Hence the packets are sampled with a probability decreasing with the order of packets and their number in a flow. Similarly to sketch-guided sampling the flow counts are maintained with a hash-based array.

Adaptive weighted sampling algorithm aims to adjust sampling rate based on observed network traffic [38]. Leveraging network traffic cycles that tend to repeat over time, the approach calculates an anticipated number of sampled packets for the next sampling interval using weighted least squares predictor. If the estimated value significantly deviates from the actual value, the approach goes through a process of adjusting sampling probability for the next interval. The adjustment is based on a set of predefined rules.

Fast filtered sampling (FFS) [32]. Similarly to other algorithms specialized for anomaly detection, FFS is designed to give a boost to small flows. Using a two-stage architecture, a sampling mechanism includes a filtering component and a traditional sampling module implementing random or uniform packet sampling. The filter maintains the counters for packets to estimate flow sizes and operates with two thresholds s and l. If a counter does not exceed s, it indicates that the flow is small enough and packets belonging to it can be passed to sampling module. If a counter reaches l, it is reset to 0, which essentially allows to 'thin' large flows before they their arrival to sampling module. Formally, the probability of a packet to be sampled Pr is defined as:

$$Pr(x) = \begin{cases} p, & \text{if } 1 \le i < s \\ ps/i & \text{if } s \le i < l \\ ps/l, & \text{otherwise} \end{cases}$$

Selective flow sampling is similar in nature to smart sampling technique, but instead of focus on 'heavy-hitters' it aims to select small flows that might be preferred for anomaly detection [3]. The probability of a flow to be sampled Pr with selecting sampling algorithm is given as follows:

$$Pr(x) = \begin{cases} \frac{z}{n \times x}, & \text{if } x > z \\ c, & \text{if } x \leq z \end{cases}$$

where x is the flow size in packets and z is a threshold indicating the size of flows considered as large. As follows from this equations, small flows are sampled with constant probability c, while those that are large are selected with probability proportional to their size. The selection of large flow can be further reduced with weighting parameter $n \geq 1$.

Adaptive traffic sampling method aims to adjust packet sampling probability to estimated flow size [20]. Dividing a sampling interval into blocks, incoming packets are sampled with a probability assessed for a previous block. At the end of each block the probability is recalculated to reflect dynamic nature of network traffic. Formally, a sampling probability P_s is calculated as $P_s = \frac{1}{1+\varepsilon^2} s$, where ε is a constant indicating a tolerance error bound and s is a total flow size, given as follows $s_h = s_{h-1} + \frac{m_h}{P_{sh}}$, m is number of sampled packets. Similar to selective sampling algorithm, adaptive traffic sampling method focuses on small flows.

6. Experiments

Data. The lack of data with application layer DoS attacks prompted us to create an evaluation dataset. We have set up a testbed environment with a victim webserver running Apache Linux v.2.2.22, PHP5 and Drupal v.7 as a content management system. The attacks were selected to represent the most common types of application layer DoS. We assume that an attacker is non-oblivious, i.e., he understands the attack, knows exactly when and how much traffic to send to maximize the attack damage. Since the main premise of low-volume DoS attacks is their ability to impact a service without significant resources on an attacker side, the attacks were generated with just enough traffic to impact the targeted service, i.e, the attacks were stopped once a server became unresponsive. As a result we noticed that to be successful it was sufficient for these attacks to produce small amounts of traffic during short periods of time.

Generated application layer DoS attacks were intermixed with the attack-free traces from the ISCX set [47]. We produced 4 types of attacks with different tools, obtaining 8 different application layer DoS attack traces. These attacks were directed towards 10 web servers in ISCX data set that have the top highest number of connections. To obtain diversity in the resulting traces, the attacks were assigned in the following manner: all eight attacks, six of them, and three of them were applied to three of these servers, the rest of the servers were assigned one of the attacks. Selections were done randomly. The details of the attacks introduced in the modified ISCX dataset are given in Table 3[2]. The resulting set contains 24 h of network traffic with total size of 4.6 GB. The overall rate for clean traffic in the modified ISCX dataset was on average 69pkt/s.

These attacks were generated using a set of existing, commonly used in the wild tools, hence we relied on parameters preset by each tool for a given type of an attack. Since we experimented with different types of attacks, the flow size was determined by a nature of an attack. The duration of an attack is the only parameter that we manipulated in our experiments, i.e., the attacks were stopped once the target server became unresponsive.

[2] The dataset is available at http://iscx.ca/appdos.

We further executed and mounted five effective attacks to the server that included flooding attack implemented with Goldeneye [45] and slow send headers attack implemented with Slowloris [44] (see Table 4) The server was immune to the rest of the attacks. The goal of these attacks was to render services on server side unresponsive while been stealthy as possible. Attacks were conducted with default parameters, which turned to be like aggressive mode and resembling flooding. We also employ tuned parameters probed to be effective with a typical web portal installation, referred as light mode. Slow send header attack did not reach necessary effect in our subject server with the light mode with only one attacker, so we employ more than one attacker to reach desire effect.

To further evaluate the proposed method on live network traces, we collected real web server traffic on an academic network for two weeks in February 2014 which amounted to 3.5 GB of data (this includes only traffic destined for port 80). The overall rate for this captured traffic was on average 2 pkt/s.During this time suspicious activity resembling application layer DoS attack was spotted. To determine the types of anomalous behavior present in these traces, the dataset was processed by Snort IDS. Although a number of attacks were detected, none of them were labeled as denial-of-service attack. We further executed and mounted five effective attacks to the server that included flooding attack implemented with Goldeneye [45] and slow send headers attack implemented with Slowloris [44] (see Table 4). Since real web server was mostly offering static content, none of the attacks requiring user interaction were applicable.

The goal of these attacks was to render services on server side unresponsive while been stealthy as possible. Attacks were conducted with default parameters, which turned to be like aggressive mode and resembling flooding. We also employ tuned parameters probed to be effective with a typical web portal installation, referred as light mode. Slow send header attack did not reach necessary effect in our subject server with the light mode with only one attacker, so we employ more than one attacker to reach desire effect. As such two out of five attacks were implemented in distributed fashion with attackers starting in short 1–3 s intervals. We use this result as a 'ground truth'.

Results. Since CUSUM algorithm is known for its sensitivity to noise, we experimented with various thresholds to determine the optimal value (see Table 5). The features used for detection algorithm were calculated over a period of 30 s. We did not find consistent size of detection time window in the studies covering application layer DoS attacks. As such we experimented with 5, 10, 15, 30, 45, 60 s time interval. With the 30 s. interval the algorithm achieved the best trade-off between time and performance. The detection effectiveness was accessed using two metrics: detection rate, DR (i.e, a percentage of attacks detected correctly), and a number of false alerts. The latter represents the plain count of alerts triggered by a detection algorithm during periods of normal traffic. Note that these alerts do not directly correspond to the number of network packets/flows.

As the results in Table 5 show, the best performance was achieved with threshold $N = 2500$ that estimates malicious increase in observed parameters. All our further experiments were conducted with this threshold. The algorithm was able to achieve perfect detection rate. It is interesting to note that although in other cases accuracy varied, there were no false positive alerts.

We also conducted several experiments on the real web traffic (Table 8). The proposed approach algorithm detected three incidents which were determined to be suspicious, although possibly not malicious, behavior resembling flooding attacks. Close analysis revealed that these incidents were botnet traversal activity.

Table 3
Data statistics of the modified ISCX dataset.

Application DoS attack	# of events	Av duration	Av # of pkts	Av. # of flows	Av. flow size (pkt)
High-volume HTTP attacks:					
DoS improved GET (Goldeneye [45])	3	452s	6084	864	7
DDoS GET(ddossim [16])	2	138s	46,081	22,103	2
DoS GET (hulk [22])	4	546s	8482	1085	8
Low-volume HTTP attacks:					
slow-send body (Slowhttptest [48])	4	834s	9106	615	15
slow send body (RUDY [43])	4	65s	7066	834	8
slow-send headers (Slowhttptest [48])	5	575s	25,503	2917	9
slow send headers (Slowloris [44])	2	150s	12,518	1881	7
slow-read (Slowhttptest [48])	2	404s	29,103	2626	11
Total:	26				

Table 4
Live web server attacks statistics.

Application DoS attack	Impact on server	Attack duration	Av. attack pkt rate
High-volume DoS	Web server becomes completely unresponsive (including all services, e.g., sql and resources CPU, memory etc.) within seconds	1 min	333pkt/s
High-volume DoS (light mode) combined with slow send headers	Web server becomes completely unresponsive after short period of time	3 min	215pkt/s
High-volume DoS (light mode) combined with slow send headers (light mode)	Web server continues to respond to clients, although at the cost of higher CPU/memory usage	1 min	180pkt/s
Slow send headers (light mode, 2 attackers)	Web server immediately becomes completely unresponsive	5 min	13pkt/s
High-volume DoS combined with slow send headers (3 attackers, light mode)	Web server becomes completely unresponsive after short period of time	1 min	346pkt/s

Table 5
Accuracy of detection (the modified ISCX data set).

CUSUM threshold	DR	Num of false alerts
$N = 2500$	100	0
$N = 3000$	96.15	0
$N = 3500$	96.15	0
$N = 4000$	88.46	0
$N = 4500$	84.61	0

7. Impact of sampling on detection

Sampling Methodology. To evaluate the impact of the sampling methods on detection of application layer DoS attacks, the proposed detection algorithm was applied to the original and sampled traces to provide adequate comparison of performance results.

The sampled traces were generated from the collected data using thirteen sampling techniques presented in Section 5. All sampling algorithms were implemented in C language. The resulting output was inspected by a detection algorithm.

The variability of sampling techniques and their parameters creates a problem when comparing the obtained detection result fairly. Since most of the detection techniques are heavily influenced by the amount of data available, it is desirable to ensure approximately equal amount of data sampled by the sampling methods. To ensure this we employ a metric proposed by Mai et al. [30] which provides a common ground to the majority of sampling techniques: *percentage of sampled flows*. As such for each of the considered sampling methods we set parameters to approximately result in equal amounts of flows, i.e., 30%, 20%, 5% and 1%. Note that due to the nature of some techniques (i.e., IP flow-based sampling, adaptive weighted sampling, and adaptive traffic sampling) obtaining the various amounts of flows was not possible.

Results with the modified ISCX data set. Table 6 summarizes the detection results with 30%, 20%, 5% and 1% of flows selected with sampling techniques. As expected the detection accuracy decreases in all cases with the decrease in the amount of data retained. Our results indicate that selective flow sampling achieves the best result among all sampling methods with 100% detection and no false positive alerts for sampling rate above 20%. This performance result obtained with 30% of flows quickly drops to 85% detection rate with 20% and further to 81% with 16% of flows (Fig. 1). The accuracy of the rest of analyzed methods drops to 80–88% range even with 30% of flows. At the same time the number of false alerts starts increasing. The only comparable performance (92% detection rate with no false positives) is obtained with sketch-guided sampling at 40% of flows. However, as the sampling rate starts decreasing, detection performance with sketch-guided sampling outperforms selective sampling. Since in many production environments sampling rate above 30% might be prohibitively expensive the use of such generic method (sketch-guided sampling) rather than specialized anomaly oriented approach might be preferred. IP flow-based sampling gives similar to sketch-guided sampling method's performance at 30% of flows. This is expected as its procedure heavily resembles sketch-guided sampling.

It should be noted that among the top five methods, only two, selective flow sampling and IP flow-based sampling, are designed for intrusion detection domain. Furthermore, adaptive weighted sampling that was specifically introduced for detection of DoS attacks (although at the network level) was able to show only 80.76% detection rate with 12 false positive alerts. In fact, with exception of selective flow sampling, all specialized sampling techniques are only able to achieve detection rate 81% (88% for IP flow-based method) with some false positives. Aiming to adjust necessary sampling parameters dynamically based on the observed network traffic, these techniques (with exception of fast filtered sampling) do not allow to manually control sampling parameters. Thus in a given network context this is the best performance that can be achieved with these methods.

Fig. 3(b) provides interesting insight into the flow distribution after sampling. In spite of variability in employed strategies, all methods at some point rely on a sampling probability that is proportional to flow size. As a result, degradation in accuracy comes from lossy and distorted nature of retained traffic clearly seen in a smoothed curve. These seven strategies (three of which

Table 6
Detection results with various sampling methods (methods designed for security domain are in **bold**). NFA stands for number of false alarms.

Sampling technique	30%		20%		5%		1%	
	DR	NFA	DR	NFA	DR	NFA	DR	NFA
Without sampling	100	0	100	0	100	0	100	0
Selective flow sampling	100	0	84.61	0	19	0	7.69	0
Sketch-guided sampling	88.46	1	84.61	7	61.53	12	23.07	0
IP flow-based sampling	88.46	2	–	–	–	–	–	–
Systematic packet sampling	84.61	15	73.07	18	38.46	8	11.53	0
Random flow sampling	80.76	0	69.23	0	34.61	5	15.38	0
Fast filter sampling	80.76	12	76.92	12	57.69	4	30.76	0
Adaptive weighted sampling	80.76	12	–	–	–	–	–	–
Adaptive traffic sampling	80.76	12	–	–	–	–	–	–
Adaptive random sampling	80.76	12	73.07	16	38.46	6	11.53	0
Random n out of N packet sampling	80.76	15	76.92	17	38.46	8	7.69	0
Random packet sampling	76.92	13	76.92	17	30.76	5	7.69	0
Sample and hold	38.46	0	7.69	0	0	0	0	0
Smart sampling	0	0	0	0	0	0	0	0

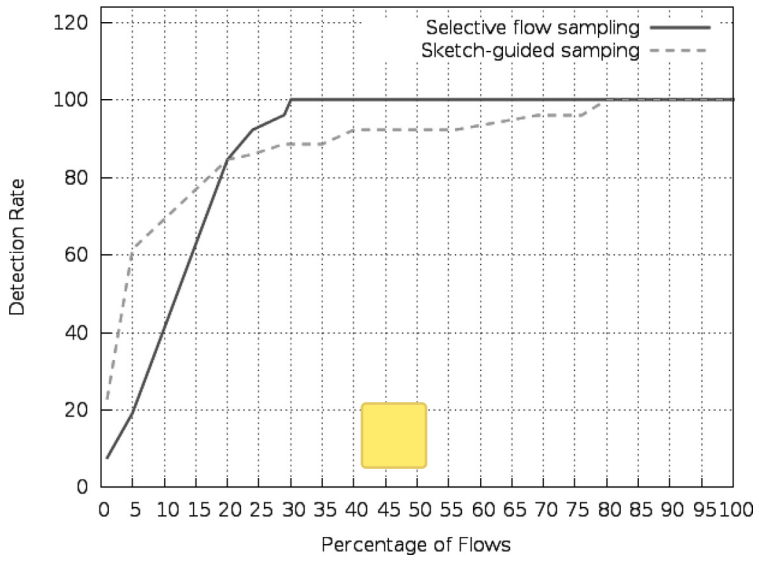


Fig. 1. Detection performance.

Table 7
Per attack flow distribution after sampling (30% of flows). Total number of attack flows retained (number of original flows) in %.

Sampling technique	Low-volume					High-volume		
	Slow body (Slowhttptest)	Slow headers (Slowhttptest)	Slow body (Rudy)	Slow headers (Slowloris)	Slow read	Ddossim	Goldeneye	Hulk
Without sampling	100%	100%	100%	100%	100%	100%	100%	100%
Selective flow sampling	98(82)	98(93)	99(93)	99(94)	99(93)	99(94)	99(93)	99(88)
Sketch-guided sampling	99.9(5)	99.9(11)	99.67(11)	99.84(19)	99.96(9)	94(75)	99.92(19)	99.79(16)
IP flow-based sampling	99.9(5)	99.99(11)	99.85(12)	99.97(20)	100(10)	94(75)	99.96(20)	100(16)
Systematic packet sampling	98(0.41)	96(0.36)	94(1)	89(1)	96(0.23)	48(24)	93(0.89)	94(1)
Random flow sampling	47(39)	46(44)	46.73(44)	46(43)	48(44)	47(44)	46(43)	46(41)
Fast filter sampling	97(0.33)	93(0.32)	93(0.90)	89(0.58)	94(0.32)	45(22)	90(0.85)	90(1)
Adaptive weighted sampling	97(0.33)	93(0.32)	93(0.90)	89(0.58)	94(0.32)	44.61(22)	90(0.85)	90(1)
Adaptive traffic sampling	97(0.33)	93(0.32)	93(0.90)	89(0.58)	94(0.32)	44.61(22)	90(0.85)	90(1)
Adaptive random sampling	97(0.33)	93(0.32)	93(0.90)	89(0.58)	94(0.32)	44.61(22)	90(0.85)	90(1)
Random n out of N packet sampling	98(0.49)	96(0.29)	95(0.87)	91(0.77)	96(0.40)	48(24)	93(0.77)	94(1)
Random packet sampling	97(0.29)	93(0.25)	93(0.81)	88(0.74)	94(0.23)	45(22.02)	89(0.66)	90 (1)
Sample and hold	14(7)	13(11)	12(9)	17(15)	10(7)	4(3)	11(10)	15(11)
Smart sampling	0.45(0.12)	0.31(0.14)	0.24(0.06)	0.13(0.03)	0.42(0.15)	0.28(0.08)	0.54(0.19)	0.37(0.12)

are designed for sampled detection of anomalies) discard any subtleties of attacks. On the other hand, top three methods (Selective flow, Sketch-guided and IP flow based) shown in Fig. 3(a) have more strategic approach to sampling aiming to reduce the flow volume while retaining the shape.

Table 7 gives more insight into an ability of these sampling methods to pick up the traces of application layer attacks. Enforcing flow shortening allows methods to retain the majority of traffic. With this however, less than 1% of the original flows remains which significantly degrades detection performance.

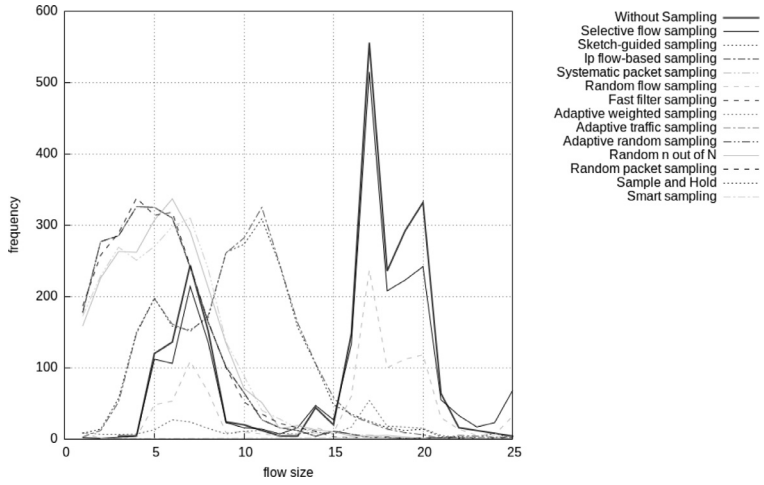
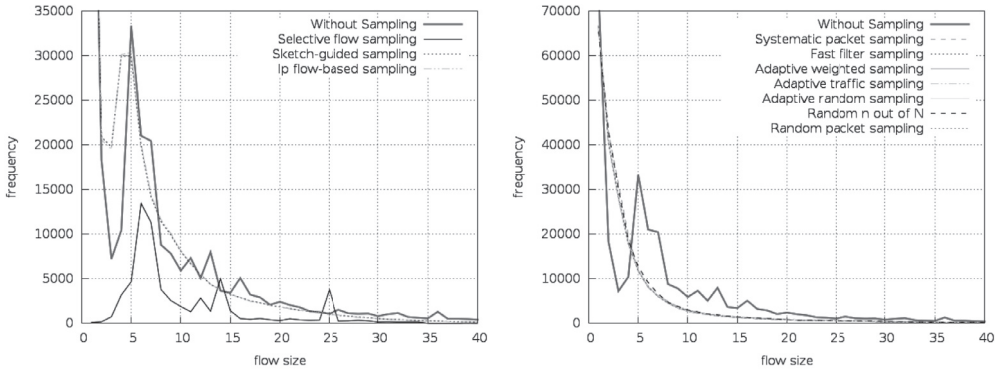


Fig. 2. Distribution of sampled attack flows with Slow body attack (Slowhttptest).



(a) Top performing methods (b) Distorted traffic

Fig. 3. Distribution of flows under various sampling schemes.

Table 8
Accuracy of detection on real web traffic.

Sampling technique	Detected incidents		Num of false alarms
	Traversal	Attacks	
Without sampling	3	5 (100%)	0
Selective flow based	0	5 (100%)	0
Sketch-guided sampling	1	5 (100%)	11
IP flow based sampling	2	5 (100%)	10

Among the worst performing methods are Sample and hold and Smart sampling. It is to some extent is an expected result as both techniques give preference to large sized flows. This is clearly seen in Fig. 3(b). Practically all small flows disappear after applying smart sampling. Sample and hold however retains small flows that even show some tendency to resemble the original traffic pattern, but the loss of small flows is so drastic that detection rate is very low (38% with 30% of flows). Since the application layer attacks are often executed with smaller and stealthier messages, focus on 'heavy-hitters' is not the most suitable strategy for these attacks' detection.

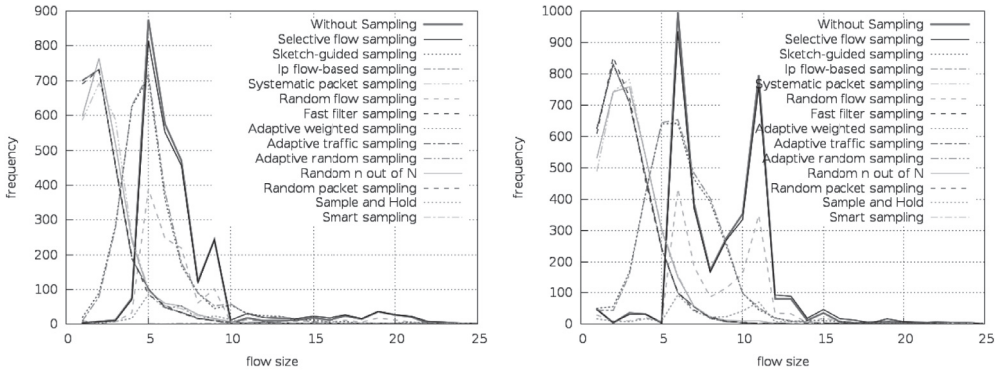
Random n-out-of-N sampling - the technique the most commonly employed in production environment (e.g., Cisco routers) - only achieved 81% detection rate. Since most of the application level DoS attacks contain flows with only a few packets (Table 3), random n-out-of-N sampling either reduces flows to a single packet flow or completely eliminates some of them, an effect that can be seen in Fig. 3(b).

A close analysis of our results show, only two types of attacks, namely flooding executed with Goldeneye [45] and Slow body executed with Slowhttptest [48], after sampling become mostly invisible to the detection. The analysis of nature of these attacks highlights the potential reasons for their stealthiness.

Detection of flooding, generally seen as volume attack (e.g., network level DoS) relies on detection of abrupt changes in traffic volume relative to global perspective. In this context even application layer high-volume attacks are often indistinguishable from the rest of traffic. As such, among flooding attacks, HTTP GET attacks generated with Goldeneye are the least noticeable with the smallest amount of generated malicious traffic (864 flows per attack). As Fig. 4 shows the largest attack volume spike is generated by small packet flows and although most of these flows are retained after sampling (see Table 7), they are heavily shortened to single-packet flows (in most cases less than 1% of original flows remains) (see Fig. 4(a)). As such slow send attack generated with RUDY although produces similar amount of small flows (834 flows per attack), exhibits different attack pattern with several spikes of flows of different size that are retained after sampling (see original flow distribution and distribution after sampling in Fig. 4(b)).

An opposite situation we observe with Slow body attack. With smallest amount of generated traffic overall (615 flows), it produces small bursts of larger sized flows. As a result sampling methods oriented on smaller flows completely distort the original traffic pattern (Fig. 2).

Since several previous studies have found other sampling techniques to give the best detection accuracy (i.e., random flow



(a) HTTP GET (Goldeneye) (b) Slow body (RUDY)

Fig. 4. Distribution of sampled attack flows.

sampling [30,31]), we experimented with the parameters of techniques aiming to determine whether we are able to achieve better results with any of the considered sampling methods. In this experimentation, among other traces we also included data traces sampled with the original parameters reported as the best in the original studies. For example, fast filter sampling was reported to provide the best false positive rate with 5% of sampled flows. Indeed, as Table 6 shows it retains this characteristic with low number of false alarms. However, with a detection rate of only 58% the lack of false positives is unlikely to be a deciding factor.

Results with real web server traffic. We also conducted several experiments on the real web traffic with the top three sampling methods, that is selective flow sampling, sketch-guided sampling and IP flow based sampling (Table 8). All sampling techniques were applied to retain 30% of flows here as well. The suspicious activity was detected only with sketch-guided and IP flow-based sampling. However, all three techniques showed good performance for detection of the actual attacks, differing only in the number of triggered false positive alerts.

8. Conclusion

In this study we looked at detection of application layer DoS attacks and proposed an efficient scheme based on nonparametric CUSUM algorithm. Using real traffic traces of application level DoS attacks we validated the effectiveness of the proposed approach. With the universal application of sampling, we further provided a thorough study of thirteen different sampling methods designed for broad traffic estimation and specifically developed for anomaly detection. We assessed how the sampling impacts application layer DoS attacks' detection and showed that even specialized sampling techniques introduce some distortion which negatively impacts detection quality.

Implication of our study. Through our analysis and experimentation we identified several factors that should be taken into account when considering detection of application layer DoS attacks in general and specifically in a context of sampling:

- **Accurate detection:** Our experiments revealed that CUSUM-based detection is an effective generic approach for detection of various types of application layer attacks.
- **Stealthiness of application-layer DoS attacks:** These attacks are generally perceived as stealthy mostly due to the fact that their traffic characteristics at the lower levels are not sufficient to distinguish them as attacks. Our experiments reveal that designing methods focused on the specific application level features increases visibility of some of these attacks to detection. Although it is possible to reduce the visibility of some of these attacks to detection by adjusting their parameters to incur a lighter impact, it consequently reduces their effectiveness nullifying their impact as attacks. In the context of sampling this however is not universal. We noticed several attacks that were generally more difficult to detect after sampling, namely HTTP GET flooding and Slow send body. Both attacks create almost no additional traffic to maintain sessions and use very few and very small flows. As a result only sampling methods that do not introduce the distortion are able to retain traces of these attacks.
- **The most suitable sampling method:** Overall, the sketch-guided sampling, a generic approach aimed at accurate traffic estimation, showed the best performance. At higher sampling rates (above 20%), selective flow sampling designed for anomaly detection achieved the best detection performance. However, as the amount of retained traffic decreased detection with the sketch-guided sampling was able to achieve significantly better accuracy. Our experiments showed that both methods introduced the least distortion closely approximating the real curve of unsampled data while shrinking the volume of retained traffic. This performance for selective flow sampling however came at the expense of high resource consumption. Since in many production environments sampling rate above 30% might be prohibitively expensive, the use of a generic sketch-guided sampling (even for detection purposes) rather than specialized anomaly oriented approach might be preferred.
- **Most sampling methods designed for anomaly detection provide low accuracy:** The methods designed specifically for anomaly detection and thus focused on small flows, namely, fast filter sampling, adaptive weighted sampling and adaptive traffic sampling, showed similar performance both in terms of accuracy and CPU utilization. Their sampling strategy caused loss of packets and flows consequently resulting in a complete distortion of traffic pattern which drastically decreased their detection rate (achieving only 81% with even 30% of flows) and increased false positives. With exception of fast filter sampling, these methods lack flexibility in adjusting sampling parameters and thus are limited in their ability to archive better results.
- **The worst methods:** Smart sampling and sample and hold methods showed the worst performance mostly due to the nature of both methods. With the heavy lean towards large flows at the expense of smaller ones, these methods are not able to retain traces of application layer attacks adversely effecting in detection accuracy. This result is consistent with the recent study conducted for general anomaly detection [30].

– **Performance of most widely employed sampling method by prominent vendors:** Random n-out-of-N sampling, one of the sampling techniques widely employed in production environments by vendors such as Cisco, Juniter, and Fortinet, also produces lossy and distorted data which negatively impacts the detection accuracy of application layer attacks. Since detection/prevention methods against these attacks are usually installed at application level, analysis of incoming traffic needs to be adjusted to compensate for distortions introduced by this sampling method.

Detection of attacks on the network level and furthermore on the application level is significantly influenced by the amount of data retained. As our results show, most of the existing sampling techniques are clearly not suitable for accurate detection of application layer DoS attacks. As such we believe that detection should be closely tied to sampling strategy to compensate for distortions introduced by sampling and to ensure better assessment of traffic characteristics.

Future work includes performing detection of application layer DoS attacks in production environment for a extended period of time. This will provide better insight into the proposed technique performance under various load conditions.

References

[1] S. Ali, I.U. Haq, S. Rizvi, N. Rasheed, U. Sarfraz, S.A. Khayam, F. Mirza, On mitigating sampling-induced accuracy loss in traffic anomaly detection systems, SIGCOMM Comput. Commun. Rev. 40 (3) (2010) 4–16.

[2] G. Androulidakis, V. Chatzigiannakis, S. Papavassiliou, M. Grammatikou, V. Maglaris, Understanding and evaluating the impact of sampling on anomaly detection techniques, in: MILCOM 2006, 2006, pp. 1–7.

[3] G. Androulidakis, S. Papavassiliou, Improving network anomaly detection via selective flow-based sampling, Commun. IET 2 (3) (2008) 399–409.

[4] Arbor Networks, Worldwide infrastructure security report, 2016, (https://www.arbornetworks.com/).

[5] H. Beitollahi, G. Deconinck, Analyzing well-known countermeasures against distributed denial of service attacks, Comput. Commun. 35 (11) (2012) 1312–1332.

[6] H. Beitollahi, G. Deconinck, Connectionscore: a statistical technique to resist application-layer ddos attacks, J. Ambi.t Intel. Human. Comput. 5 (3) (2014) 425–442.

[7] S. Bhatia, D. Schmidt, G. Mohay, Ensemble-based ddos detection and mitigation model, in: SIN '12, ACM, 2012, pp. 79–86.

[8] D. Brauckhoff, B. Tellenbach, A. Wagner, M. May, A. Lakhina, Impact of packet sampling on anomaly detection metrics, in: IMC '06, ACM, New York, NY, USA, 2006, pp. 159–164.

[9] E. Brodsky, B. Darkhovsky, Nonparametric Methods in Change Point Problems, Springer Science & Business Media.

[10] W.O. Chee, T. Brennan, HTTP POST, 2010, (http://goo.gl/xYXh1v).

[11] Y. Chen, K. Hwang, W.-S. Ku., Collaborative detection of DDoS attacks over multiple network domains, IEEE Trans. Parallel Distrib. Syst. 18 (12) (2007) 1649–1662.

[12] B.-Y. Choi, J. Park, Z.-L. Zhang, Adaptive random sampling for load change detection, in: SIGMETRICS '02, ACM, New York, NY, USA, 2002, pp. 272–273.

[13] A. Chonka, Y. Xiang, W. Zhou, A. Bonti, Cloud security defence to protect cloud computing against HTTP-DoS and XML-DoS attacks, J. Netw. Comput. Appl. 34 (4) (2011) 1097–1107.

[14] CISCO, Sampled netflow, 2013, (http://goo.gl/uLi8vD).

[15] K.C. Claffy, G.C. Polyzos, H.-W. Braun, Application of sampling methodologies to network traffic characterization, in: SIGCOMM '93, ACM, New York, NY, USA, 1993, pp. 194–203.

[16] DDOSIM, Layer 7 DDOS simulator, 2010, (http://sourceforge.net/projects/ddosim).

[17] N. Duffield, C. Lund, M. Thorup, Charging from sampled network usage, in: IMW '01, ACM, New York, NY, USA, 2001, pp. 245–256.

[18] N.G. Duffield, C. Lund, M. Thorup, Learn more, sample less: control of volume and variance in network measurement., IEEE Trans. Inf. Theory 51 (5) (2005) 1756–1775.

[19] C. Estan, G. Varghese, New directions in traffic measurement and accounting: focusing on the elephants, ignoring the mice, ACM Trans. Comput. Syst. 21 (3) (2003) 270–313.

[20] X. He, Y. Wu, Q. Wang, An adaptive traffic sampling method for anomaly detection, in: ICICSE 2009, 2009, pp. 142–146.

[21] N. Hohn, D. Veitch, Inverting sampled traffic, IEEE/ACM Trans. Netw. 14 (1) (2006) 68–80.

[22] HULK, Http unbearable load king, 2012, (http://goo.gl/PWhEJk).

[23] J. Jung, B. Krishnamurthy, M. Rabinovich, Flash crowds and denial of service attacks: characterization and implications for cdns and web sites, in: WWW '02, ACM, New York, NY, USA, 2002, pp. 293–304.

[24] Juniper, Juniper flow monitoring, 2013, (http://goo.gl/LL0zNf).

[25] A. Kumar, J.J. Xu, Sketch guided sampling - using on-line estimates of flow size for adaptive data collection, INFOCOM, 2006.

[26] G. Macia-Fernandez, J. Diaz-Verdejo, P. Garcia-Teodoro, Mathematical model for low-rate DoS attacks against application servers, IEEE Trans. Inf. Forens. Security 4 (3) (2009) 519–529.

[27] G. Maciá-Fernández, J.E. Díaz-Verdejo, P. García-Teodoro, Assessment of a vulnerability in iterative servers enabling low-rate DOS attacks, in: ESORICS'06, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 512–526.

[28] G. Maciá-Fernández, J.E. Díaz-Verdejo, P. García-Teodoro, Evaluation of a low-rate DoS attack against iterative servers, Comput. Netw. 51 (4) (2007) 1013–1030.

[29] G. Maciá-Fernández, R.A. Rodríguez-Gómez, J.E. Díaz-Verdejo, Defense techniques for low-rate DoS attacks against application servers, Comput. Netw. 54 (15) (2010) 2711–2727.

[30] J. Mai, C.-N. Chuah, A. Sridharan, T. Ye, H. Zang, Is sampled data sufficient for anomaly detection? in: IMC '06, ACM, New York, NY, USA, 2006, pp. 165–176.

[31] J. Mai, A. Sridharan, C.-N. Chuah, H. Zang, T. Ye, Impact of packet sampling on portscan detection, Select. Areas Commun. IEEE J. 24 (12) (2006) 2285–2298.

[32] J. Mai, A. Sridharan, H. Zang, C.-N. Chuah, Fast filtered sampling, Comput. Netw. 54 (11) (2010) 1885–1898.

[33] M. Mehra, M. Agarwal, R. Pawar, D. Shah, Mitigating denial of service attack using CAPTCHA mechanism, in: ICWET '11, ACM, New York, NY, USA, 2011, pp. 284–287.

[34] MOD Security, What can ModSecurity do?, 2016, (https://modsecurity.org/).

[35] G. Mori, J. Malik, Recognizing objects in adversarial clutter: breaking a visual CAPTCHA, in: CVPR, vol. 1, 2003, pp. 134–141.

[36] S.Y. Nam, T. Lee, Memory-efficient IP filtering for countering DDoS attacks, in: APNOMS'09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 301–310.

[37] Q. Pan, H. Yong-feng, Z. Pei-feng, Reduction of traffic sampling impact on anomaly detection, in: ICCSE 2012, 2012, pp. 438–443.

[38] A. Patcha, J.-M. Park, An adaptive sampling algorithm with applications to denial-of-service attack detection, in: ICCCN 2006, 2006, pp. 11–16.

[39] Prolexic, Quaterly global DDoS attack report, 2013, (http://www.prolexic.com/).

[40] M.Z. Rafique, M.A. Akbar, M. Farooq, Evaluating dos attacks against sip-based voip systems, in: Proceedings of the 28th IEEE Conference on Global Telecommunications, in: GLOBECOM'09, IEEE Press, Piscataway, NJ, USA, 2009, pp. 6130–6135.

[41] S. Ranjan, et al., Ddos-resilient scheduling to counter application layer attacks under imperfect detection, in: In Proceedings of IEEE INFOCOM, 2006, pp. 23–29.

[42] S. Ranjan, R. Swaminathan, M. Uysal, A. Nucci, E. Knightly, DDoS-shield: DDoS-resilient scheduling to counter application layer attacks, IEEE/ACM Trans. Netw. 17 (1) (2009) 26–39.

[43] R. Raz, RUDY: universal HTTP DoS - are you dead yet?, 2010, (http://goo.gl/74DoBG).

[44] RSnake, Slowloris HTTP DoS, 2009.

[45] J. Seidl, GoldenEye layer 7 DoS test tool, 2012.

[46] sFlow, Configuring fortigate appliances, 2009, (http://goo.gl/SgfRtK).

[47] A. Shiravi, H. Shiravi, M. Tavallaee, A.A. Ghorbani, Toward developing a systematic approach to generate benchmark datasets for intrusion detection, Comput. Security 31 (3) (2012) 357–374.

[48] Slowhttptest, Application layer DoS attack simulator, 2013, (https://code.google.com/p/slowhttptest/).

[49] SpiderLabs, Mitigation of Apache range header DoS attack, 2011, (http://goo.gl/uC6dGK).

[50] Y. Xie, S. zheng Yu, A novel model for detecting application layer ddos attacks, in: IMSCCS '06, 2, 2006, pp. 56–63.

[51] Y. Xie, S.-Z. Yu, Monitoring the application-layer DDoS attacks for popular websites, IEEE/ACM Trans. Netw. 17 (1) (2009) 15–25.

[52] Y. Xuan, I. Shin, M. Thai, T. Znati, Detecting application denial-of-service attacks: a group-testing-based approach, IEEE Trans. Parallel Distrib. Syst. 21 (8) (2010) 1203–1216.

[53] C. Ye, K. Zheng, Detection of application layer distributed denial of service, in: Computer Science and Network Technology (ICCSNT), 2011 International Conference on, vol. 1, 2011, pp. 310–314.

[54] J. Yu, Z. Li, H. Chen, X. Chen, A detection and offense mechanism to defend against application layer DDoS attacks, ICNS 2007, 2007. 54–54.

Hossein Hadian Jazi is Senior Cyber Threat Intelligence Advisor at Bell Canada. He did his Master's Studies in Malware Analysis at the Canadian Institute of Cyber Security, University of New Brunswick, Canada. He is an active researcher and his research interests include network security and malware analysis. Currently his focus is on hunting and analyzing cyber threats in Canadian land-escape.

Hugo Gonzalez is a PhD student at the Canadian Instute for Cybersecurity, University of New Brunswick, Canada. He is a faculty member of the Polytechnic University of San Luis Potosi, Mexico. Hugo's current research interests include network security, malware analysis and machine learning. Hugo is a member of the Association for Computing Machinery, the IEEE Computer Society and The Honeynet Project.

Dr. Natalia Stakhanova is the New Brunswick Innovation Research Chair in Cyber Security at the University of New Brunswick, Canada. She completed her Ph.D. work in Network Security in 2007 at Iowa State University, USA. Natalia's work revolves around building secure systems and includes intrusion detection, malicious software, and security evaluation and assessment. Natalia Stakhanova was the recipient of the Nokia Best Student Paper Award at The IEEE International Conference on Advanced Information Networking and Applications (AINA). She chaired and served on the program committee of several conferences and workshops in area of information security and assurance, including the Conference on Privacy, Security and Trust (PST). Natalia developed a number of technologies that have been adopted by high-tech companies such as IBM and she currently has three patents in the field of computer security.

Dr. Ghorbani has held a variety of positions in academia for the past 36 years and is currently the Canada Research Chair (Tier 1) in Cybersecurity, the Dean of the Faculty of Computer Science, and the Director of the Canadian Institute for Cybersecurity. His current research focus is cyber security, Web Intelligence, and Critical Infrastructure Protection. Dr. Ghorbani is the co-inventor on 3 awarded patents in the area of Network Security and Web Intelligence and has published over 200 peer-reviewed articles during his career. He has supervised over 165 research associates, postdoctoral fellows, graduate and undergraduate students during his career. His book, Intrusion Detection and Prevention Systems: Concepts and Techniques, was published by Springer in October 2010. He is one of the co-founders of the Privacy, Security, Trust (PST) Network in Canada and its international annual conference. In 2007, Dr. Ghorbani received the University of New Brunswick's Research Scholar Award. Dr. Ghorbani has developed a number of technologies that have been adopted by high-tech companies. He co-founded two startups, Sentrant and EyesOver in 2013 and 2015, respectively. Dr. Ghorbani is the co-Editor-In-Chief of Computational Intelligence Journal. He was twice one of the three finalists for the Special Recognition Award at the 2013 and 2016 New Brunswick KIRA award for the knowledge industry.