A Survey on Slow DDoS Attack Detection Techniques

Oluwatobi Shadrach Akanji
Department of Computer Science
Federal University of Technology
Minna, Nigeria
akanjioluwatobishadrach@yahoo.com

Opeyemi Aderiike Abisoye
Department of Computer Science
Federal University of Technology
Minna, Nigeria
o.abisoye@futminna.edu.ng

Sulaimon A. Bashir
Department of Computer Science
Federal University of Technology
Minna, Nigeria
bashirsulaimon@futminna.edu.ng

Oluwaseun Adeniyi Ojerinde
Department of Computer Science
Federal University of Technology
Minna, Nigeria
o.ojerinde@futminna.edu.ng

***Abstract*— The ease with which DDoS attack is being launched using publicly available tools has made DDoS to be a recurring security problem. However, given the immense work by researchers to stem the tide of volumetric DDoS, attackers have resorted to using a slow DDoS attack which is similar to benign traffic thus making detection and mitigation difficult. This paper seeks to provide the scholarly community with a survey on slow DDoS attack detection techniques worked upon by researchers over time. A low amount of work has been done when the work on slow DDoS detection is juxtaposed with that of volumetric DDoS. However, researchers who have worked on detecting slow attacks have achieved remarkable results. Machine learning detection technique has proven to be effective with random forest and K-Nearest Neighbour (KNN) being the major algorithms that have consistently achieved good results in terms of Area Under Curve (AUC), accuracy, and false positive rate. Other detection techniques of time series and performance model have also been effective against slow DDoS but need to be improved upon given the non-linearly separable nature of a slow attack and benign traffic. Most researchers resorted to using attack tools to generate attack data due to the absence of a standard data set. Recommendations for future studies include exploration of detecting slow table overflow attacks in SDN before a table overflow event occurs.**

Keywords—Slow DDoS, Slowloris, Slow POST, Slow Read, Slow attack detection, Slow HTTP

I INTRODUCTION

The threats to devices in a networked environment keeps on metamorphosing because of the variety in network devices, protocols, and configuration. Among these threats is Distributed Denial of Service (DDoS). DDoS attacks involve the use of a large number of Internet-enabled and connected devices to synchronously send illegitimate requests to a target thus overwhelming the target's capacity to respond to the requests [1]. The manipulation of data transfer rates which consequently consumes the target's resources is one of the strategies used to cause a DDoS. A DDoS situation is reached when the attacker maintains connections or sends data to the victim which results in the unavailability or improper functioning of the services offered by the victim to legitimate users. According to [2], volumetric and application layers are the major categories of DDoS attacks. Volumetric attacks are characterized by large data transfer rate launched against the targets which exhaust the bandwidth of the target's links or the memory storage and processing power of the target. Unlike volumetric attacks structured on the network and transport layers, the application layer attacks exploit the behaviour of application layer protocols thereby increasing detection difficulty and circumventing network and transport layers DDoS detection mechanisms. The application layer attacks could employ either fast or slow data transfer rate to achieve DDoS. The use of slow or low data transfer rate to achieve application layer DDoS requires establishing and maintaining connections with the victim for prolonged periods hence, hindering service availability to legitimate clients. Slow data transfer rate DDoS are also known as slow DDoS.

Slow DDoS attacks are generally application layer attacks that exploit application layer protocols of HTTP, FTP, IMAP, and SMTP. Unlike volumetric DDoS, it utilizes less bandwidth and small computational resource of the attacker [2]. The low bandwidth usage characteristic of slow DDoS enables it to evade detection because the data transfer rate bears semblance with that of either a legitimate user with a slow connection or one whose device has low data transmission capacity [3]. The attacker occupies most or all the service queues at the application level thereby causing incoming requests to be discarded [4]. Slow HTTP DDoS, an attack against web servers, is the most prominent in this category which can be attributed to the vast amount of web servers. A slow HTTP DDoS attacker establishes a connection with the webserver using the three-way handshake protocol after which the connection is maintained using a few amount of data [5]. Although it is true in some situations that slow DDoS attacks focus on slow data transfer rate, it also entails the use of few amount of data relative to the bulk of data requested for or being transferred to sustain a connection to the victim [4][6]. The advantages of the slow DDoS attacks which includes detection evasion, low attack resource requirement, and easy configuration endears it to DDoS attackers. Also, the ability to launch a slow HTTP DDoS attack from a mobile phone has compounded the problem of detection and mitigation given the wide use of mobile phones for Internet connectivity [7]. In general, these DDoS attacks are aimed at targets such as OpenFlow switches, web servers, file servers, and mail servers.

II. TYPES OF SLOW ATTACKS

Classification of slow DDoS attacks is based on either the application layer or the device an attack targets. The types of slow DDoS attacks with their targets are examined in this section.

a. Slow HTTP DDoS

A slow HTTP DDoS is a type of DDoS which exploits the way the HTTP protocol on web servers operates particularly the lack of time-bound active connection rules and the need to wait for the completion of requests [8][9]. HTTP is one of the most popular Internet protocols which executes on the TCP/IP protocol suite, the backbone of the World Wide Web (WWW). Whenever a HTTP request is sent to a web server, the request is accompanied with a header which contains information such as window size, protocol version, and window scaling necessary for the webserver to process the request and send the required response appropriately [9]. To launch a slow HTTP DDoS attack, a normal TCP or UDP connection is first established with the victim and then the attacker seeks to maintain every connection established by either sending or reading few amounts of bytes to or from the webserver. There are three types of slow HTTP DDoS: the slow read, the slow POST, and the slow GET. To hide the attack origin, the attacker may utilize HTTPS as the transport protocol to establish and maintain connections [9].

Slow Read

A slow read DDoS attack is aimed at causing the unavailability of web services to legitimate clients by requesting for data resource from the web server and then forcing the victim to send the reply at a low rate [5][10]. After establishing a connection with the webserver, the attacker requests for a resource while advertising a small TCP window. The TCP window defines the number of bytes readable by a client. The attacker ensures that the TCP window advertised is smaller than the web server's buffer size thus causing delays which fills up the webserver's buffer with responses waiting for dispatch [11]. In some cases, the attacker advertises a TCP window size of 0 bytes which makes the web server wait indefinitely for the client to be available for response receipt, however, timeout mechanisms and zero-byte window detection mechanisms implemented on the webserver makes the attack easy to detect. Hence, attackers may resort to using varying amount of bytes large enough to sustain the connection and evade detection mechanisms but small enough to cause a DDoS scenario [12]. The attacker continues to establish numerous connections to the webserver until it has occupied most or all of the available connections on the webserver. This ensures that there is an increase in the web server's response time or availability of the web service to legitimate clients is none existent. The method of operation for a slow read DDoS attacker is illustrated in fig. 1.

Slow POST

Unlike the slow read attack, the slow POST attack sends data to the webserver at a rate that maintains the connections established for a long period. The slow POST attack is also known as the *R-U-Dead-Yet* (RUDY) or slow body attack relies on sending a HTTP POST request which advertises a large content-header value. On receiving the request, the target server allocates resources necessary for the completion of the data transfer until the connection is completed or terminated by the client [11]. Since the webserver waits, as long as the connection is active until the specified length of data is received, the attacker resorts to sending small amounts of data to the server at intervals, regular or random, smaller than the timeout value of inactive connections. For the attack to be successful, the attacker launches several similar connections to the webserver and initiates the same data transfer method [13][9]. For instance, an attacker might have advertised a content-length of 5 megabytes (MB) for a POST request to a web server but sends about 20 to 30 kilobytes (KB) within the range of 15 to 25 seconds given that the timeout value for inactive connections on the webserver is 30 seconds. Accordingly, it will take approximately 2,500 seconds or 41 minutes per connection to complete such a request. Illustrated in fig. 2 is the slow POST attack process.

Slow GET

Similar to the slow POST attack, the slow GET attack also involves sending of data to the target web server. Slow GET attacks are also known as a slow header or slowloris attacks. A legitimate GET request is sent to the webserver after establishing a connection, however, on receiving a 200 OK message from the server which indicates that the server is ready to receive the headers, the attacker splits the header into several chunks which are sent at a low rate. In a normal scenario, the header consists of two Carriage-Return Line-Feed (CRLF) characters ("\r\n\r\n") which signify the end of the header and the beginning of the body to the webserver thus allowing the webserver to begin processing of the request. A single CRLF character signifies the end of a line and the beginning of another in the header request [14]. However, in an attack scenario, both CRLF characters are not transmitted thus causing the victim web server to keep the connections open as it waits indefinitely for the completion of the header requests [9][15]. The indefinite wait of the server causes the dropping of connection requests made by legitimate clients because the connection limit of the webserver has been reached. Slow GET attack description is shown in fig. 3.

Slow TCAM

The emergence of Software-Defined Networking (SDN) brought about the decoupling of the control and data planes into different devices thus allowing for a centralized view of the network. The controller of the network resides in the control plane as it maintains a unified view of the network while the switches reside in the data plane operating as packet forwarding devices. The switch maintains a Ternary Content Addressable Memory (TCAM) where it stores all the flow rules obtained from the controller whenever a new flow arrives at the switch. However, the TCAM has its limit as it can store rules from 1500 to 3000 flow rule entries [16]. A slow TCAM attack sends new flows to a switch thereby triggering flow rule requests from the switch and installation of the flow rules to the switch by the controller. The flow rules are then maintained by sending small amounts of data at intervals less than the TCAM inactive flow rule timeout value. An attacker seeks to establish numerous flow rules on the switch aimed at causing new flows from legitimate traffic to be dropped since the TCAM reaches its maximum amount of flow rules allowed and the flow rules in the switch are still active. This type of slow attack can be made effective through the recruitment of a large number of bots that send new flows to the switch at a low rate.

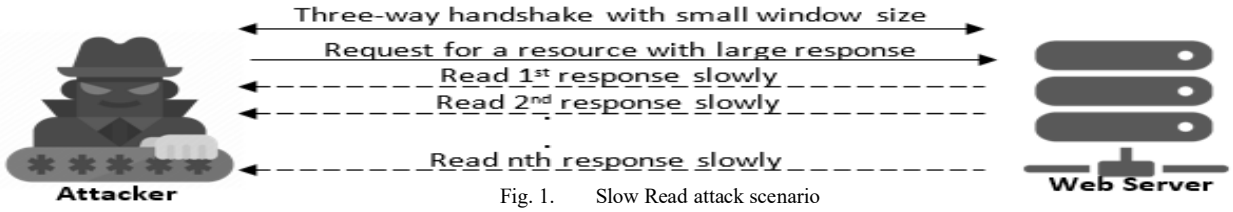

Fig. 1. Slow Read attack scenario

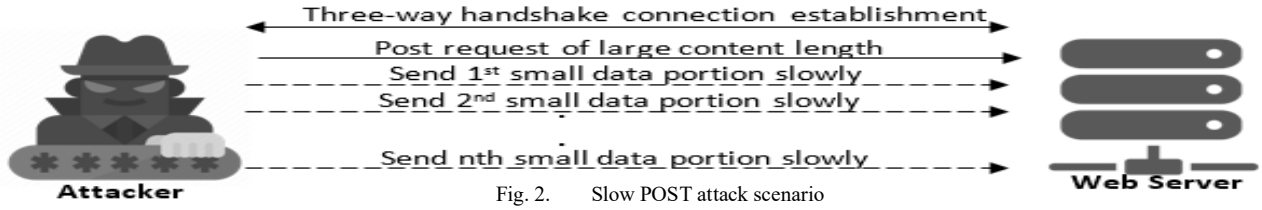

Fig. 2. Slow POST attack scenario

In fig. 4, the attacker makes an initial connection to a web server connected to the target switch in the SDN network. The initial network is then sustained by transferring data at a rate that evades any flow entry timeout mechanism set. Furthermore, the attacker increases the number of connections that passes through the switch until it exhausts the flow entry capacity of the switch. As illustrated in fig. 4, the limit of flow entries in the switch is m unique connections whereas the attacker attempts to make n unique connections where n is greater than m. This invariably leads to a table overflow on the target switch.

III DETECTION METHODS

The detection of slow DDoS attacks is difficult because the behaviour of the attack is similar to that of a slow client that sends legitimate traffic. Also, since the attacker establishes a connection to the webserver by adhering to legitimate connection rules in the case of slow HTTP DDoS or sends new flows to the SDN switch requesting for a resource in the SDN network in the case of slow TCAM, attack detection is challenging. Slow DDoS detection methods proposed by researchers can be classified into machine learning, time series, probability with distance metric, and performance models techniques. Detection techniques that employ machine learning methods seek to predict the class category of a new flow record or packet-based on previously identified records of benign and attack traffic or based on the similarity observed between previous traffic. The use of time series is aimed at harnessing the function of time progression to detect an attack. The possibility of traffic to be an attack traffic is considered using probability-based measurements. Similarly, distance-based measurements compute the possibility of a new traffic to be an attack traffic based on the closeness of the features of the new traffic to that of a previously established attack traffic. Since an attack changes the state and behaviour of a web server, performance model technique of attack detection calculates the behaviour of the webserver or data transfer rate under normal circumstances and seeks to identify any behaviour that deviates from the initially established behaviour. Table I presents a summary of the detection techniques with their strengths and weaknesses.

b. Machine Learning

Machine learning techniques of supervised and unsupervised learning were used in detecting slow DDoS attacks. Machine learning techniques under the supervised learning category which makes predictions based on previously observed features is the most prominent category used in the analysis.

The use of 5-NN, Naïve Bayes, multilayer perceptron, support vector machines, JRIP, Random forest, C4.5 decision trees, and logistic regression to detect DoS attacks of slow POST and slowloris was evaluated in [13]. The learners achieved high Area Under Curve (AUC) which was attributed in part to the use of Netflow feature set. The highest AUC value of 0.99905 with a class ratio of 50:50 was recorded in RF and the second highest AUC of 0.99904 with a class ratio of 65:35 was recorded in RF. Although their work showed good detection of slow POST and slowloris attacks, they employed the use of a DoS attack that originates from a source. Since DoS attacks are easier to detect compared to DDoS due to the variation in features such as source and destination IP address pair, the work charts a path for further research using DDoS. Furthermore, the similarity in the way slow POST and slowloris attacks are launched might have lent some degree of high detection rate to their experiment. However, their work buttresses the findings of other researchers about the random forest being a good machine learning technique to detect slow and volumetric DDoS. Also, 5-NN achieved high detection rate compared to other techniques used in their work however, it was surpassed by random forest.

Six classifiers of random forest, KNN, logistic regression, SVM, decision trees, and deep neural networks were used in [17] to detect slow HTTP attacks. KNN and Decision trees achieved high detection rates. KNN had an accuracy of 99.81%, false positive rate of 0.08%, and false negative of 1.09% while decision tree achieved an accuracy of 99.87%, false positive of 0%, and false negative rate of 0.03% when

there was an equal composition of attack and legitimate traffic in the dataset. The achievement of KNN strengthens the view that KNN, an unsupervised learning algorithm, can be used to detect slow attacks. However, the detection time of KNN when an unbalanced dataset was used was 61.21 seconds which means that prompt detection of slow attacks when KNN is used is not always guaranteed.

Instead of using full packet captures, Netflow features were used in [8] with eight classifiers to detect slow read attacks in SDN networks. The use of Netflow was attributed to the low packet processing overhead associated with Full Packet Captures (FPC). The classifiers of random forest, C4.5 N, 5-Nearest Neighbour, C4.5D, MLP, JRip, SVM, and Naïve Bayes achieved an AUC of 96.76%, 96.72%, 96.69%, 96.62%, 95.06%, 94.71%, 89.22%, and 88.94% respectively.

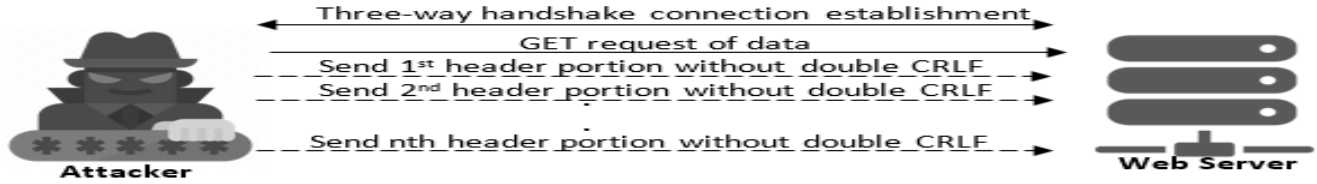

Fig. 3. Slow GET attack scenario

Fig. 4 Slow TCAM attack scenario

Since high AUC reflects high TPR and low FPR, the random forest is seen as the best classifier for detecting slow read attacks. Here, random forest classifier proves to be the best classifier that detects slow read attacks.

Detection of slowloris and slow POST attacks in encrypted traffic by clustering extracted features and performing machine learning detection of anomalies was performed in [18]. Machine learning techniques used are single linkage clustering, k-means, fuzzy c-means, self-organizing maps, and DBSCAN. K-means, fuzzy c-means, and self-organizing maps achieved high detection rates of 99.9957% with detection rate.

In another work, machine learning techniques to detect RUDY attacks using features from bi-directional network instances false positive rate of 0.0043% for slowloris attacks. Also, K-means, fuzzy c-means, and self-organizing maps achieved high detection rates of 99.9931% with false positive rate of 0.0043% for slow post attacks. Kmeans, another unsupervised learning algorithm, achieves high

selected using an ensemble feature selection approach containing 10 different feature ranker methods aimed at extracting the most important features for the detection of RUDY attacks at the network level. It was observed that the usage of fewer features increases detection time and analysis accuracy. SANTA dataset that was obtained from the network of a commercial Internet Service Provider (ISP) together with the RUDY attack dataset obtained during pen-testing used in their work. Three classification methods of K-Nearest Neighbor (K-NN) where k is five and two forms of C4.5 decision trees (C4.5D and C4.5N) were used to build the predictive models. The selected features include features that represent three main characteristics of traffic size, packet similarity, and traffic velocity. When seven features were used, results obtained for the AUC metric shows that 99.83%, 99.96%, and 99.99% were achieved by C4.5N, C4.5D, and 5-NN respectively; for true positive rate, 99.07%, 98.90%, and 98.97% were achieved by C4.5N, C4.5D, and 5-NN respectively; and for

false positive rate, 0.029%, 0.041%, and 0.0265% were achieved by C4.5N, C4.5D, and 5-NN. When all the features were used, the AUC metric achieved results of, 99.88%, 99.40%, and 99.99% by C4.5N, C4.5D, and 5-NN respectively; for true positive rate, 98.73%, 98.66%, and 98.83% were achieved by C4.5N, C4.5D, and 5-NN respectively; and for false positive rate, 0.0282%, 0.0307%, and 0.0316% were achieved by C4.5N, C4.5D, and 5-NN respectively. The higher AUC value means higher TPR and lower FPR. 5-NN also achieves a good detection by having the highest AUC and the lowest FPR values when seven features were selected. The increase in AUC and FPR values in 5-NN when all the features were used points the effect of large feature usage on the detection rate of 5-NN. Although higher AUC was obtained, the corresponding increase in false positive rate cannot be substantiated given that the clustering algorithm flags a greater amount of legitimate traffic as malicious [19].

Usage of the random forest algorithm to detect slow read attacks in a cloud environment was performed in [12]. Raw TCP logs of a slow read attack were analysed and preprocessed before passing the data to the random forest classifier. The accuracy of the random forest classifier increases with an increase in the number of trees however, the computational complexity also increases. Pre-pruning of the trees has proven to increase the false negative rate to 50.10% compared to 1.90% when pre-pruning was not used. Accuracy of 83.34% was recorded when pre-pruning was used compared to 99.37% when pre-pruning was not used. However, 0% false positive rate was observed in either case. The use of pre-pruning of trees in random forest makes the solution not to be developed appropriately through the growth of the trees. The absence of pre-pruning sheds more light on the reason random forest classifiers have consistently shown its suitability in detecting slow attacks. It was also noted in their work that increasing the number of trees to improve performance gain may not be justified when the number of trees reaches a point where the computational cost of finding a solution affects the detection rate adversely.

HTTP count and delta time were used in [20] with other features to detect slow HTTP attacks using machine learning classifiers of naïve bayes, naïve bayes multinomial, multilayer perceptron, random forest, logistic regression, and radial basis function network. Results obtained indicate that naïve bayes multinomial has the best accuracy of 93.67%, true positive of 91.49%, and false positive of 3.10% compared to the results obtained for other machine learning techniques.

Detection of slow attacks using machine learning techniques has proven that although detection might be difficult, it is not impossible. The ability to detect slow attacks rely on the correct identification and tweaking of the classifier's parameters. As observed in KNN, using the value of K as five gives better result compared to other values of K. Also, the use of pre-pruning has been shown to affect random forest classification adversely.

c. Time Series

Detection of slow POST, header, and read DoS attacks based on a nonparametric CUSUM algorithm was introduced in [21]. It detects changes in the distribution of observed values. 13 different sampling techniques were used. Detection rate reduces as the threshold number increases. The threshold of 2500 achieved 100% detection rate with 0% false alerts. Selective flow sampling achieved the highest detection rate when the sampling rate is greater than 20%. The result obtained using selective flow sampling can be attributed to the selection of small flows for analysis rather than large flows. This ensures that the slow attacks that generate small flows are easily identified.

The use of spectral analysis to detect low rate DoS that affect Apache 2.2 servers was worked on in [22]. The spectral analysis is focused on the distribution of power over the frequency of a time series. In their work, a Discrete Fourier Transform was used to transform the signal to the frequency domain. It was observed that the beginning of an attack is more detectable than an ongoing attack using their method. Different detectability using different bot wait times was noticed as wait times also affect detectability. It was observed that detection using spectral analysis was possible when the attacker used fixed waiting times or floods the server with connection requests when starting the attack.

Time series decomposition that separates the time series into random and trend components on which the cumulative sum (CUSUM) technique and double autocorrelation technique were applied respectively in the work by [23]. Detection latency of 32 seconds was recorded with FPR and FNR of 4.3% and 9.8% respectively.

Time series method of detecting slow DDoS attacks have achieved good detection rate however, it is worthy of note that several factors affect the detection rate adversely compared to machine learning techniques.

d. Probabilistic with Distance-based Similarity Metric

Euclidean distance similarity metric was employed to detect slow attacks in [24]. The analysis of log files to calculate the similarity was used. Another distance similarity metric, Hellinger distance, was used in [25] to measure the distance between the probability distributions of the normal and attack traffic generated. Evasion of the detection system is inevitable if an attacker can generate packets whose probability distribution is similar to that of the normal traffic used as a benchmark.

Chi-square statistics was also used to detect slow rate DoS attacks. Selecting the appropriate threshold and interval time proved difficult as an increase in the interval time improves recall rate and causes a high false positive rate too but a reduction in interval time reduces recall rate and improves false positive rate [26].

The use of probability and distance-based similarity metric has not proven to be effective in detecting slow DDoS attacks yet. It can be attributed to the non-linearity of the attack type in contrast with volumetric attacks. Volumetric attacks are easily detected because the deviation of its features from benign traffic features is immense. The dilemma of using probability-based detection is evident in [26].

e. Performance Model

Packet inter-arrival time and window size analysis were used in [11] to detect slow HTTP DDoS attacks. It was identified that the average window size in client to server communication for normal traffic, slowloris, RUDY, and slow read attack are 34041, 14123, 14034, and 7241 respectively while in server to client communication the average window size recorded was 27022, 6854, 6856, and 0 respectively. It can be observed that the average values of slowloris and RUDY attacks are closer to each other which can be attributed to the similarity of their attack. It was also recorded that the average packet delta time in client to server for normal, slowloris, RUDY, and slow read attacks were 302.28, 75.16, 74.123, and 339.28 ms respectively while that of the server to the client was 151.12, 0.115, 0.561, and 28.759 ms.

A solution to slow HTTP DDoS attacks on OpenStack cloud platform was implemented in [15]. A packet pre-monitoring module identifies the behaviour of packets and the passes it to the classifier zone module. An allowed and blocked list is also maintained. All clients are placed in the allowed list

until the client violates some conditions. The average network delay is calculated by sending 5 pings to the client and the average reply response time is calculated by taking into account the time the client responds to the ping messages. Once the delay between the HTTP requests exceeds five times the calculated network delay, the client is moved to the block list zone. The five times network delay is based on considering the processing time for applications. Frequent advertisement of TCP window of zero is monitored and placed in the block list. Furthermore, POST or GET requests sent to the webserver when 80% of the timeout value has elapsed are treated as an attacker and placed in the block list. In their work, slow body attacks were detected when connection requests reached 1700 while slow read attacks were detected when connection requests reached 1000.

Connection threshold that aids in detecting a slow attack in an SDN network was examined in [27]. A slow attack is detected when an incomplete HTTP request is made when the number of open connections on the web server exceeds the predetermined threshold number of concurrent connections being processed.

The TABLE FULL message generated in SDN when new rules cannot be installed due to a full TCAM was utilized in [16] to detect a TCAM attack which in turn activates a mitigation mechanism.

Reverse proxy was used in [28] to mitigate slowloris attack and detect the attack by measuring the stress at the server. The reverse proxy handles requests on behalf of the original server pending the completion of the request.

TABLE I. SUMMARY OF SLOW DDOS DETECTION TECHNIQUES

S/N	Author	Detection Technique	Strength and result	Weakness
1	Calvert and Khoshgoftaar [13]	Machine Learning	The high detection rate of 99.905% in random forest	Only DoS was examined Only slow header and slow POST were examined The high computational cost for generating trees
2	Siracusano *et al.* [17]	Machine Learning	Decision Tree accuracy of 99.87%	A small change in data can cause immense change in optimal solutions
3	Kemp *et al.*[8]	Machine Learning	Use Netflow for low packet processing overhead Random Forest had AUC of 96.76%	Only slow read was examined
4	Zolotukhin *et al.* [18]	Machine Learning	K-means achieved a detection rate of 99.9931%	Only slowloris and slow POST were examined
5	Najafabadi *et al.*[19]	Machine Learning	5-NN achieved AUC of 99.99% with false positive of 0.0265%	Only slow POST attack was examined
6	Shafieian *et al.*[12]	Machine Learning	Random forest without pre-pruning achieved 99.37% accuracy with 1.90% false negative, and 0% false positive	Only slow read was examined Tree creation computational cost
7	Singh and De [20]	Machine Learning	Naïve Bayes multinomial achieved an accuracy of 93.67%	A high false positive rate of 3.10% was recorded
8	Jazi *et al.*[21]	Time Series	Selective flow sampling achieved the highest detection rate of 100% when set to a sampling rate greater than 20%.	High resource consumption due to sampling rate
9	Brynielsson and Sharma [22]	Time Series	Detects the beginning of an attack	Attack wait times affected detectability The continuation of an attack may not be detected
10	Liu and Kim [23]	Time Series	Average attack detection time of 32 seconds	False positive rate of 4.3% and false negative rate of 9.8%
11	Cusack and Tim [24]	Probability with Distance-based Similarity	Low processing overhead	Detection of attack after havoc has been caused due to the use of log files for analysis
12	Tripathi *et al.* [25]	Probability with Distance-based Similarity	Simple probability distributions and Hellinger distances were utilized to detect attacks	Possibility of detection evasion by generating attack packets with probabilities close to the normal traffic probabilities
13	Tripathi and Hubballi [26]	Probability with Distance-based Similarity	0% false positive rate recorded for ΔT = 5 minutes 100% recall rate recorded for ΔT = 20 and 25 minutes	Large ΔT increases false positive rate but improves recall rate and low ΔT reduces recall rate but improves the false positive rate
14	Muraleedharan and Janet [11]	Performance model	Identification and recording of core features that signifies any of the slow HTTP attacks	Only DoS attacks were examined
15	Idhammad *et al.* [15]	Performance model	Effective in identifying slow connection masqueraders	Variable window size and data transfer interval small enough to cause DDoS can circumvent the detection technique
16	Hong *et al.* [27]	Performance model	Simple to implement	Difficulty establishing an appropriate threshold
17	Dantas *et al.* [16]	Performance model	Ease of implementation	The attack is detected only after the table overflow has occurred
18	Yeasir *et al.* [28]	Performance model	Ease of detection because attacks stress the server's resources	Only slowloris attack was considered
19	Shtern *et al.* [29]	Performance model	Ease of detection by using the performance of the webserver to identify attacks	The dilemma of when to establish the performance metric to be used for comparison

The establishment of a performance model using the central processing unit (CPU) utilization and time, workload, disk utilization and time, waiting time and throughput to form a baseline that signifies attack was explored by [29]. However, the dilemma of when to establish the baseline is an obstacle identified in their model. Perhaps, the baseline created might have been performed when an attack was taking place which makes it difficult to detect subsequent attacks easily using that established baseline.

Performance-based models of detecting slow DDoS attacks have proven to be good in detecting attacks however, they are not devoid of issues as evident in [29]. Selecting the appropriate threshold has been difficult to perform given the dynamic nature of an attack and benign traffic.

III. DISCUSSION

As shown in Table I, machine learning detection techniques have proven to be effective and efficient in detecting slow DDoS attacks in computer networks. Eight works of literature on slow DDoS detection using machine learning were examined. Prominent among the supervised and unsupervised learning categories are the random forest and KNN techniques respectively. However, the computational overhead of random forest and the slow detection time of KNN in the presence of unbalanced datasets are their shortcomings.

Performance models and time series techniques of detecting slow attacks trail behind machine learning techniques as evident in Table I in terms of results achieved and an improvement in their approach of detecting DDoS attacks is needed. Six performance model technique and three time series detection technique were evaluated. The advantage these aforementioned techniques have over machine learning techniques are their ability to extract features easily, perform detection faster, and conserve the detection system's resources because they do not rely on external modules to detect attacks.

The use of probability with distance-based similarity technique has not yielded any remarkable result yet. Although three research works were identified, the studies either encountered hitches of either possible attack detection circumvention or inability to detect the attack appropriately. This is due, in part, to the inability to represent slow attack traffic using concrete values.

All the techniques used in detecting slow DDoS attacks as examined in this study have shortcomings however, the use of machine learning detection technique offers more prospect in detecting attacks than any of the other techniques studied.

IV. CONCLUSION

As observed, research into the field of slow DDoS attack detection is low compared to that of volumetric attacks. This could be attributed to the ease with which researchers can perform experiments that detect volumetric attacks without resorting to other techniques of feature extraction and technologies that may be beyond their scope. This lack of adequate research on detecting slow attacks is also evident through the absence of standard datasets of slow attacks compared to that of volumetric DDoS which has CAIDA, NSL-KDD, and DARPA datasets. In the absence of the dataset, slow DDoS attack researchers have resorted to creating their dataset by simulating the attack using tools such as slowHTTPTest, slowloris.py, and OWASP switchblade amongst others.

For further studies, researchers can develop a standard data set for slow attacks and also improve upon performance model and time series detection techniques. Also, the adjustment of parameters in machine learning techniques together with feature selection should be explored. Furthermore, studies on slow table overflow attack detection and mitigation are needed given that detecting and mitigating a table overflow attack after it has wreaked havoc is not efficient and proactive.

In summary, although some researchers were able to demonstrate how slow DDoS attacks can be detected, more needs to be done in the field of slow DDoS attack detection and mitigation considering its detection difficulty, low attack resource usage, and the ability to launch one from a mobile phone.

REFERENCES

[1] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "SOFTWARE-DEFINED NETWORKING SECURITY : PROS AND CONS," no. June, pp. 73–79, 2015.

[2] R. Swami, M. Dave, and V. Ranga, "Software-defined Networking-based DDoS Defense Mechanisms," *ACM Comput. Surv.*, vol. 52, no. 2, p. 36, 2019.

[3] J. Boite, P. A. Nardin, F. Rebecchi, M. Bouet, and V. Conan, "Statesec: Stateful monitoring for DDoS protection in software defined networks," in *2017 IEEE Conference on Network Softwarization: Softwarization Sustaining a Hyper-Connected World: en Route to 5G, NetSoft 2017*, 2017.

[4] E. Cambiaso, G. Papaleo, G. Chiola, and M. Aiello, "Slow DoS attacks: definition and categorisation," *Int. J. Trust Manag. Comput. Commun.*, vol. 1, no. 3/4, p. 300, 2013.

[5] J. Park, "Analysis of Slow Read DoS Attack and Countermeasures on Web servers," *Int. J. Cyber-Security Digit. Forensics*, vol. 4, no. 2, pp. 339–353, 2015.

[6] E. Cambiaso, G. Papaleo, and M. Aiello, "Slowcomm: Design, development and performance evaluation of a new slow DoS attack," *J. Inf. Secur. Appl.*, vol. 35, pp. 23–31, 2017.

[7] P. Farina, E. Cambiaso, G. Papaleo, and M. Aiello, "Understanding DDoS Attacks From Mobile Devices," 2015.

[8] C. Kemp, C. Calvert, and T. M. Khoshgoftaar, "Utilizing netflow data to detect slow read attacks," in *Proceedings - 2018 IEEE 19th International Conference on Information Reuse and Integration for Data Science, IRI 2018*, 2018, pp. 108–116.

[9] S. Suroto, "A Review of Defense Against Slow HTTP Attack," *JOIV Int. J. Informatics Vis.*, vol. 1, no. 4, p.

127, 2017.

[10] D. Ameyed, F. Jaafar, and J. Fattahi, "A slow read attack using cloud," in *Proceedings of the 2015 7th International Conference on Electronics, Computers and Artificial Intelligence, ECAI 2015*, 2015, pp. SSS33–SSS38.

[11] N. Muraleedharan and B. Janet, "Behaviour analysis of HTTP based slow denial of service attack," in *Proceedings of the 2017 International Conference on Wireless Communications, Signal Processing and Networking, WiSPNET 2017*, 2018, vol. 2018-Janua, pp. 1851–1856.

[12] S. Shafieian, M. Zulkernine, and A. Haque, "CloudZombie: Launching and detecting slow-read distributed denial of service attacks from the Cloud," in *Proceedings - 15th IEEE International Conference on Computer and Information Technology, CIT 2015, 14th IEEE International Conference on Ubiquitous Computing and Communications, IUCC 2015, 13th IEEE International Conference on Dependable, Autonomic and Se*, 2015, pp. 1733–1740.

[13] C. L. Calvert and T. M. Khoshgoftaar, "Impact of class distribution on the detection of slow HTTP DoS attacks using Big Data," *J. Big Data*, 2019.

[14] O. Yevsieieva and S. M. Helalat, "Analysis of the Impact of the Slow HTTP DoS and DDoS Attacks on the Cloud Environment," p. 5, 2017.

[15] M. Idhammad, K. Afdel, and M. Belouch, "Detection System of HTTP DDoS Attacks in a Cloud Environment Based on Information Theoretic Entropy and Random Forest," *Secur. Commun. Networks*, vol. 2018, 2018.

[16] Y. G. Dantas, I. E. Fonseca, and V. Nigam, "Slow TCAM Exhaustion DDoS Attack," vol. 1, pp. 17–31, 2017.

[17] M. Siracusano, S. Shiaeles, and B. Ghita, "Detection of LDDoS Attacks Based on TCP Connection Parameters," in *Global Information Infrastructure and Networking Symposium*, 2018.

[18] M. Zolotukhin, T. Hamalainen, T. Kokkonen, and J. Siltanen, "Increasing web service availability by detecting application-layer DDoS attacks in encrypted traffic," in *2016 23rd International Conference on Telecommunications, ICT 2016*, 2016.

[19] M. M. Najafabadi, T. M. Khoshgoftaar, A. Napolitano, and C. Wheelus, "RUDY attack: Detection at the network level and its important features," in *Proceedings of the 29th International Florida Artificial Intelligence Research Society Conference, FLAIRS 2016*, 2016, pp. 282–287.

[20] K. J. Singh and T. De, *Emerging Research in Computing, Information, Communication and Applications*. 2015.

[21] H. H. Jazi, H. Gonzalez, N. Stakhanova, and A. A. Ghorbani, "Detecting HTTP-based application layer DoS attacks on web servers in the presence of sampling," *Comput. Networks*, vol. 121, pp. 25–36, 2017.

[22] J. Brynielsson and R. Sharma, "Detectability of low-rate HTTP server DoS attacks using spectral analysis," in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2015*, 2015, pp. 954–961.

[23] H. Liu and M. S. Kim, "Real-time detection of stealthy DDoS attacks using time-series decomposition," in *IEEE International Conference on Communications*, 2010.

[24] B. Cusack and Z. Tian, "Detecting and tracing slow attacks on mobile phone user service," in *Proceedings of the 14th Australian Digital Forensics Conference, ADF 2016*, 2016, no. December, pp. 4–10.

[25] N. Tripathi, N. Hubballi, and Y. Singh, "How Secure are Web Servers? An empirical study of Slow HTTP DoS attacks and detection," in *Proceedings - 2016 11th International Conference on Availability, Reliability and Security, ARES 2016*, 2016, pp. 454–463.

[26] N. Tripathi and N. Hubballi, "Slow rate denial of service attacks against HTTP/2 and detection," *Comput. Secur.*, vol. 72, pp. 255–272, 2018.

[27] K. Hong, Y. Kim, H. Choi, and J. Park, "SDN-Assisted Slow HTTP DDoS Attack Defense Method," *IEEE Commun. Lett.*, vol. 22, no. 4, pp. 688–691, 2018.

[28] M. Yeasir, M. Morshed, and M. Fakrul, "A Practical Approach and Mitigation Techniques on Application Layer DDoS Attack in Web Server," *Int. J. Comput. Appl.*, vol. 131, no. 1, pp. 13–20, 2015.

[29] M. Shtern, R. Sandel, M. Litoiu, C. Bachalo, and V. Theodorou, "Towards mitigation of low and slow application DDoS attacks," in *Proceedings - 2014 IEEE International Conference on Cloud Engineering, IC2E 2014*, 2014, no. Vm, pp. 604–609.