JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGY
http://e-journal.uum.edu.my/index.php/jict

How to cite this article:
Akanji, O. S., Abisoye, O. A., & Iliyasu, M. A. (2021). Mitigating slow hypertext transfer protocol distributed denial of service attacks in software defined networks. *Journal of Information and Communication Technology, 20*(3), 277-304. https://doi.org/10.32890/jict2021.20.3.1

Mitigating Slow Hypertext Transfer Protocol Distributed Denial of Service Attacks in Software Defined Networks

[1]Oluwatobi Shadrach Akanji,
[2]Opeyemi Aderiike Abisoye
& [3]Mohammed Awwal Iliyasu
[1,2&3]Department of Computer Science, Federal University of Technology Minna, Nigeria

akanji.pg914786@st.futminna.edu.ng
o.abisoye@futminna.edu.ng
awwaliliyasu@gmail.com

Received: 23/7/2020 Revised: 26/12/2020 Accepted: 5/1/2021 Published: 11/6/2021

ABSTRACT

Distributed Denial of Service (DDoS) attacks have become one of the persistent forms of attacks on information technology infrastructure connected to public networks due to the ease of access to DDoS attack tools. Researchers have been able to develop several techniques to curb volumetric DDoS, which overwhelm the target with a large number of request packets. However, a limited number of research has been executed on mitigating slow DDoS. Attackers have resorted to slow DDoS because it mimics the behaviour of a slow legitimate client, thereby causing service unavailability. This paper provides the scholarly community with an approach to boost service availability

in web servers under slow hypertext transfer protocol (HTTP) DDoS attacks through attack detection. Genetic Algorithm and Support Vector Machine (SVM) were selected to facilitate attack mitigation in a software-defined networking environment simulated in GNS3. Genetic Algorithm was used to select the NetFlow features, which indicated the presence of an attack and also determined the appropriate regularisation parameter, C, and gamma parameter for the SVM classifier. The results obtained showed that the classifier had detection accuracy, area under the receiver operating curve, true-positive rate, false-positive rate, and false-negative rate of 99.89 percent, 99.89 percent, 99.95 percent, 0.18 percent, and 0.05 percent respectively. Furthermore, the algorithm for subsequent implementations of the selective adaptive bubble burst mitigation mechanism was presented. This study contributes towards the ongoing research in detecting and mitigating slow HTTP DDoS attacks with emphasis on the use of machine learning classification and meta-heuristic algorithms.

Keywords: Genetic Algorithm, Slow DDoS mitigation, Slow Distributed Denial of Service, Software Defined Network, Support Vector Machine.

INTRODUCTION

Distributed Denial of Service (DDoS) attacks are assaults against information technology infrastructure using Internet-enabled and connected devices to synchronously send requests to the victim at a rate that overwhelms the processing capacity and response rate of the victim (Dabbagh et al., 2015). Exhaustion of the victim's resources is the aim of the attack such that the resources that would have been used to serve requests from legitimate clients would be occupied by the attack requests, thus denying the legitimate clients access to the resource (Swami et al., 2019a). This results in the availability of the service, an aspect of network security among the confidentiality, integrity, and availability triad, being tampered with (Swami et al., 2019b). A large portion of DDoS attacks reported are volumetric DDoS attacks that send large numbers of requests at a rate faster than the victim can process. However, several techniques to detect and curb volumetric attacks have been developed, one of which includes routing the traffic through the Cloudflare network that detects and defends the target from the attack (Ezekiel et al., 2017). Due to the ease with which

volumetric DDoS attacks easily trigger control measures, attackers have resorted to using a form of DDoS that occupies resources using traffic that resembles the way a legitimate client would request for resources (Cambiaso et al., 2017; Jaafar et al., 2019; Muraleedharan & Janet, 2018). This form of DDoS is known as slow DDoS.

The study by Cambiaso et al. (2013) found that slow DDoS or low-rate attacks are hard to detect, which exploit the operation of the application layer. It exploits application layer protocols such as Hypertext Transfer Protocol (HTTP), simple mail transfer protocol, file transfer protocol, and Internet message access protocol by behaving either as a legitimate client sending traffic over a slow connection or one with low response processing capacity (Dhanapal & Nithyanandam, 2019; Suroto, 2017; Swami et al., 2019a). Slow DDoS causes service unavailability by occupying all or most connections to the victim and sustaining the connection for a long time by sending data to the victim over the connections. The data used to sustain the connection is usually large enough to prevent the closure of the connection by the victim but small enough to cause the elongation of the time to complete sending the request or receiving the response (Cambiaso et al., 2013). Unlike volumetric DDoS that saturates the bandwidth of the victim's network link, slow DDoS has low bandwidth usage of the victim's network link, thus requiring lesser attack resources (Cambiaso et al., 2013; Shtern et al., 2014; Suroto, 2017). With a large number of web services on the Internet, attackers use a variation of slow DDoS, known as slow HTTP DDoS, to target web servers.

Slow HTTP DDoS attack is a type of slow DDoS targeted at web servers by exploiting HTTP protocol's mode of operation. This attack is launched after a Transmission Control Protocol (TCP) connection has been established with the victim web server (Idhammad et al., 2018; Tayama & Tanaka, 2018). There are three variations of slow HTTP DDoS: slow HTTP header, slow HTTP POST, and slow read DDoS attack. Slow HTTP header DDoS attacks, also known as slow GET attacks, send HTTP GET messages to the web server without transmitting two carriage return and line feed characters that signifies the end of the GET request. Therefore, the request from the client is not concluded, which makes the web server wait indefinitely for the completion of the request before processing of the header can begin (Idhammad et al., 2018; Muraleedharan & Janet, 2018; Suroto, 2017).

Slow HTTP POST DDoS attacks use the *Content-Length* field in the header to inform the web server of a large data transfer. However, instead of sending the data at once, the attack focuses on sending the data in small chunks, thus prolonging the connection to the web server (Idhammad et al., 2018; Swami et al., 2019b). Unlike slow HTTP header and POST DDoS attacks that are based on data transmission from the attacker to the web server, slow read attacks are based on data transmission from the web server to the attacker. The slow read DDoS attacker requests for a resource on the server that has large data to be transmitted but adjusts the TCP window of the attack machine to force the web server to send the data in small bytes, thus prolonging the connection time. Slow HTTP DDoS entails making multiple connections to the web server at a time or at intervals so as to seize available connections on the web server. To protect the web server from any form of DDoS attack, effective detection techniques are employed including the use of machine learning algorithms.

The field of machine learning, a sub-field of artificial intelligence, focuses on developing machines that can learn from previous experiences based on the data provided to make decisions in the future (Latah & Toker, 2019). Machine learning is classified into four groups based on the common learning method between the learning algorithms, namely supervised, unsupervised, reinforcement, and semi-supervised learning (Agarwal, 2014). Supervised learning uses predefined knowledge to determine the class, for example, of a new dataset. Unlike supervised learning, unsupervised learning relies on the relationships established among the dataset provided to make decisions on an unseen dataset (Agarwal, 2014). Supervised machine learning algorithms, such as Decision Tree, Artificial Neural Network, and SVM, have been used in classifying traffic into normal and attack categories with high accuracy, but results in increased computational cost. To improve the performance of a machine learning algorithm, meta-heuristic algorithms such as Particle Swarm Optimisation (PSO), Ant Colony Optimisation, and Genetic Algorithm are applied for feature selection from the dataset or tuning of the machine learning algorithm's parameter. Detecting the attack is not sufficient to protect the attack victim unless measures to mitigate the attack are triggered.

Mitigation of the slow HTTP DDoS attack refers to the use of methods that prevent service degradation or resource exhaustion on the web

server when an attack is detected by halting or diminishing the rate of attack (Jaafar et al., 2019; Yeasir et al., 2015). Previous volumetric and slow DDoS attack mitigation techniques include redirecting the traffic to a verifying device as identified in the work by Beigi-Mohammadi et al. (2017), Lukaseder et al. (2018), and Schehlmann and Baier (2013); limiting the rate of attack, which was used by Bhunia and Gurusamy (2017) and Yuan et al. (2017); dropping the attack traffic selectively as explored by Fonseca and Nigam (2016); or spreading the traffic to replicas of the attack victim (Ameyed et al., 2015; Sattar et al., 2016). In view of these mitigation strategies, a global view of the network status and traffic that traverse the network is needed for effective detection and mitigation of slow HTTP DDoS (Lukaseder et al., 2018).

Software-Defined Networking (SDN) solves the absence of a unified network view, management, and flexible device configuration by fusing logically centralised network management with network programmability through the separation of the data plane from the control plane (Benzekki et al., 2016; Polat & Polat, 2021). SDN is comprised of a network controller and switches. The controller, operating at the control plane, governs the manner of data forwarding, whereas the switch, operating at the data plane, receives and forwards data based on rules defined by the controller (Dabbagh et al., 2015). Collection of traffic data is performed by the controller through the OpenFlow protocol. However, traffic data generation is not lightweight in OpenFlow when compared to NetFlow because the controller needs to request for the data at intervals from the switch. In NetFlow, the traffic data are exported without the request-response overhead (Hamad et al., 2016; Schehlmann & Baier, 2013). NetFlow is a technology of Cisco Systems that monitors network traffic and exports the network flows. Network flow refers to the unidirectional network packet stream between source and destination applications according to Schehlmann and Baier (2013), which offers efficient storage of network packets by grouping them into flow summaries (Kemp et al., 2018). Using NetFlow for traffic collection in SDN also reduces packet processing overhead as compared to Full Packet Captures (FPC) (Kemp et al., 2018).

In this work, a dataset of attacks and legitimate traffic from NetFlow version 5 is generated to record exports in a simulated SDN network

in GNS3. Genetic Algorithm is then used to select the appropriate features that determine the status of the traffic being examined through the selection of regularisation and gamma parameters for the SVM classifier. The SVM classifier is executed on the final dataset using the selected regularisation and gamma parameters to generate a model for real-time detection of attacks. Furthermore, the selective adaptive bubble burst mitigation mechanism is proposed. This paper offers a six-fold contribution. First, the various slow DDoS attack detection techniques are organised into categories. Second, the various methods of mitigating DDoS attacks in SDN are identified. Presentation of the NetFlow features that determine the presence of attack traffic as selected by the Genetic Algorithm constitutes the third contribution. For the fourth contribution, the regularisation and gamma parameters of the radial basis function kernel-based SVM classifier are identified. Finally, this paper proposes the algorithm of a DDoS mitigation technique known as selective adaptive bubble burst and presents the results of the slow DDoS detection phase. This paper is structured as follows: related work on SDN-based slow DDoS attack detection and SDN with non-SDN based slow HTTP DDoS attack mitigation techniques; a methodology that presents the SVM and Genetic Algorithm used to select features and detect the slow HTTP DDoS attacks; presentation of the results obtained in the feature selection, SVM parameter tuning, and slow HTTP DDoS attack detection; and finally, areas of further studies are identified and the conclusion is presented.

RELATED WORK

SDN-Based Slow DDoS Attack Detection Techniques

Detection of slow DDoS attacks has proven to be difficult as compared to volumetric DDoS because its traffic is similar to the traffic of a slow legitimate client or a client accessing the resource through a low bandwidth network. Researchers have worked at solving this problem using various methods that can be categorised into machine learning, performance model, probability with distance metric, and time series. Detection of slow attacks using machine learning utilises previous data regarding normal and attack traffic to determine the category of future traffic to be analysed. Similarly, a performance model

detection technique establishes a model of resource utilisation on a web server when it is not under attack and classifies any deviation from the established model as an attack scenario. The function of time progression with probability measurements is used in time series detection technique to identify an attack scenario. Moreover, the probability with distance metric detection technique harnesses the closeness of the current traffic to previously known attack traffic.

1) Machine Learning Detection Techniques

Supervised and unsupervised machine learning techniques are used in detecting slow DDoS attacks. The prominent category used among the machine learning categories is the supervised learning technique. Machine learning algorithms from supervised and unsupervised learning techniques were used in the study by Calvert and Khoshgoftaar (2019) to detect Denial of Service (DoS) attacks of slow POST and header (slowloris). 5-NN, JRIP, Random Forest, Multilayer Perceptron (MLP), Naïve Bayes, SVM, logistic regression, and C4.5 Decision Tree were the learning algorithms employed in the work. A high area under the receiver operating curve (AUC) was recorded that was partly attributed to the use of NetFlow for traffic collection. In the normal to attack traffic class ratio distribution of 50:50, Random Forest achieved the highest AUC of 0.99905 among other class distributions and learning algorithms. Although the work achieved a good detection rate, DoS attack was the basis of the work, which entailed that the attack originated from a machine, thus making attack detections easier considering the lack of variation in attack behaviour. Furthermore, the slow attacks examined, i.e. slow POST and slowloris, were similar in mode of execution. However, their work strengthened the finding of other researchers regarding Random Forest as a good machine learning algorithm for detecting slow and volumetric DDoS.

Perez-Diaz et al. (2020) examined the use of SDN to mitigate low-rate DDoS while performing attack detection using machine learning classifiers. In their work, the Open Networking Operating System (ONOS) controller operating in a Mininet virtual machine hosted the Intrusion Prevention System (IPS), while an external Intrusion Detection System (IDS) was used. Communication between ONOS and IDS was through an identification application programming

interface. J48, Random Tree, REP Tree, Random Forest, MLP, and SVM were the machine learning classifiers used and accuracy values of 90.68 percent, 91.76 percent, 90.37 percent, 94.41 percent, 95.01 percent, and 93.10 percent, respectively were achieved. Here, MLP achieved better accuracy as compared to Random Forest; however, the result showed the significance of Random Forest in detecting low-rate DDoS. Although the accuracy of SVM was on the average as compared to the other classifiers, it indicated the prospects of being viable in detecting low-rate DDoS attacks.

Detection of slow HTTP attacks using K-Nearest Neighbour (K-NN), SVM, logistic regression, Random Forest, Decision Tree, and deep neural network was examined by Siracusano et al. (2018). High detection accuracy of 99.87 percent and 99.81 percent were achieved by Decision Tree and K-NN, respectively. This signified that K-NN, an unsupervised learning algorithm, can be used in detecting slow attacks effectively. However, in comparison to Decision Tree, the detection time of K-NN was poor with only 61.21 seconds of attack detection time. Slow read attack detection was evaluated by Kemp et al. (2018) using Random Forest, 5-NN, MLP, SVM, JRip, Naïve Bayes, C4.5D, and C4.5N. In their work, Random Forest, C4.5N, 5-NN, and C4.5D achieved high AUC values of 96.76 percent, 96.72 percent, 96.69 percent, and 96.62 percent, respectively. Here, Random Forest proved to be good in detecting slow read attacks.

Zolotukhin et al. (2016) examined the detection of slowloris and slow POST attacks in encrypted traffic. Single linkage clustering, k-means clustering, fuzzy c-means, self-organising maps, and DBSCAN were the machine learning techniques used for detection. A high detection rate of 99.9957 percent with a false-positive rate of 0.0043 percent for slowloris attacks was recorded in k-means, self-organising maps, and fuzzy c-means. For slow POST attacks, k-means, self-organising maps, and fuzzy c-means achieved a high detection rate of 99.9931 percent with a false-positive rate of 0.0043 percent. Slow POST (RUDY) attacks were also examined via the SANTA dataset by Najafabadi et al. (2016) using 5-NN, C4.5N, and C4.5D to develop predictive models. The results obtained for the AUC metric showed that 5-NN, C4.5N, and C4.5D achieved 99.99 percent, 99.83 percent, and 99.96 percent, respectively. In terms of false-positive rate, 5-NN, C4.5N, and C4.5D achieved 0.0316 percent, 0.0282 percent, and

0.0307 percent, respectively. These results were achieved due to the feature selection process that was applied to the dataset to identify features of importance of which seven features were selected. It was observed that an increase in the number of features resulted in an increase in AUC and false positive values. Since false positive increased, it indicated that that more legitimate traffic was flagged as malicious.

TCP logs of slow read attacks were used by Shafieian et al. (2015) to detect slow read attacks that occur in the cloud environment using the Random Forest classifier. It was observed that although an increase in the number of trees caused an increase in accuracy, computational complexity increased too. Pre-pruning of trees resulted in an increase in false negative value to 50.10 percent as compared to when pre-pruning was not used, which had a value of 1.90 percent. An accuracy of 99.37 percent was recorded when pre-pruning was not used as compared to 83.34 percent obtained when pre-pruning was used. Radial basis function network, logistic regression, MLP, Naïve Bayes, and Naïve Bayes Multinomial were used by Singh and De (2015) to detect slow HTTP attacks using features including HTTP count and delta time. Naïve Bayes Multinomial obtained the best results with an accuracy of 93.67 percent, false positive of 3.10 percent, and true positive of 91.49 percent as compared to other machine learning techniques evaluated in the work.

As observed, detecting slow attacks, either DDoS or DoS, depends on how well the machine learning algorithm's parameters are well-tuned to perform the detection task. Using a value of 5 in K-NN yields better results as compared to other values of K. Furthermore, feature selection influences the detection rate as identified in Najafabadi et al. (2016).

2) Performance Model Detection Techniques

Analysis of the window size and packet inter-arrival times was used to detect slow HTTP DDoS attacks (Muraleedharan & Janet, 2018). The average window size in client to server communication for normal traffic, slowloris, RUDY, and slow read attack as observed were 34,041, 14,123, 14,034, and 7,241, respectively while in server to client communication, the values recorded were 27,022, 6,854, 6,856,

and 0, respectively. The similarity in attack mode between slowloris and RUDY was evident in the closeness of average window size value.

Idhammad et al. (2018) proposed the monitoring of average network delay using five ping requests, frequent advertisement of TCP window size of zero, and POST or GET requests sent to the server when 80 percent of the timeout had elapsed. In their work, slow read attacks were spotted when 1,000 connection requests were reached and slow body attacks were detected when connection requests reached 1,700. Establishing a connection threshold by monitoring the number of open connections on the web server against a predefined threshold of concurrent connections in processing was performed in Hong et al. (2018) to determine slow HTTP attacks. A slow HTTP attack was detected when incomplete HTTP requests exceeded the connection threshold.

Monitoring of the TABLE FULL message generated by an SDN switch when the Ternary Content Addressable Memory (TCAM), the flow rule storage location, was full was used to detect slow TCAM attacks in Dantas et al. (2017). Once a TCAM attack was detected, the mitigation mechanism that purged flow rules based on a predefined algorithm became active.

Measurement of the stress on the web server was employed by Yeasir et al. (2015) to detect slow HTTP attacks. Once the attack was detected, a reverse proxy mechanism would handle all subsequent incoming traffic for the primary web server. Shtern et al. (2014) proposed a performance model based on central processing unit (CPU) utilisation and time, workload, throughput, waiting time, and disk utilisation of the web server to signify the presence of attack traffic. However, the period to establish the baseline became a problem because it might be established when an attack was in progress or when all traffic used cases that could not be captured.

3) Probability with Distance Metric Detection Technique

Analysis of log files to establish similarity using Euclidean distance similarity metric was employed by Cusack and Tian (2016). Hellinger distance, a distance similarity metric, was used in Tripathi et al. (2016) to measure the distance between the probability distributions of the

attack and normal traffic generated. Detection evasion was expected if an attacker could create packets whose probability distribution was similar to that of the normal traffic used as a standard.

Tripathi and Hubballi (2018) used Chi-square statistics to detect slow rate DoS attacks. An appropriate selection of threshold and interval time was proved to be difficult. Reduction in interval time reduced recall rate and improved false-positive rate. Nevertheless, there was an increase in the interval time that improved recall rate and caused a high false-positive rate. The dilemma of using probability-based detection was evident in their work due to the lack of concrete distinguishing factor between slow attacks and normal traffic as compared to volumetric attacks.

4) Time Series Detection Technique

Jazi et al. (2017) applied nonparametric cumulative sum (CUSUM) algorithm to detect slow header, POST, and read DoS attacks. 13 sampling techniques that detected changes in the distribution of observed values were used. It was noted that as the threshold number increased, the detection rate reduced. A detection rate of 100 percent was recorded when the threshold value reached 2,500. When the rate of sampling exceeded 20 percent, selective flow sampling achieved the highest detection rate.

Brynielsson and Sharma (2015) proposed detecting low-rate attacks on Apache 2.2 servers using spectral analysis. Spectral analysis was aimed at the distribution of power over the frequency of a time series. Discrete Fourier transform was used in their work to transform the signal to the frequency domain. Attack detection was easier when it started than if it were ongoing. Using spectral analysis for detection was possible when the attacker sent a large number of connection requests when the attack began or when the fixed waiting time was applied.

False positive and negative rates of 4.3 percent and 9.8 percent, respectively, with detection latency of 32 seconds were recorded by Liu and Kim (2010). Decomposition of time series that separated the time series into trend and random components was used in the detection of stealthy DDoS on which the double autocorrelation

technique and CUSUM technique were applied. In comparison to machine-learning detection techniques, time series methods had low detection results. Nevertheless, it proved to be better than probability with distance-based similarity metric detection method.

SDN-Based Attack Mitigation Techniques

Mitigating slow read DDoS attacks to ensure high availability of the web server's resources was explored by Ameyed et al. (2015). In their approach, a failure isolation zone that ensured high availability and redundancy in the cloud was implemented by distributing traffic between web servers in two zones. Once the maximum number of connections was reached on the first zone, new connections were redirected to the second zone and slow connections in the first zone were deleted. The effectiveness of their approach was not implemented. The strength of the ModSecurity mitigation mechanism was evaluated against slow read DDoS attacks by Park (2015). Establishment of a threshold for the number of connections from an Internet Protocol (IP) address was used to detect and trigger the mitigation mechanism. However, with an increased number of attackers whose number of established connection was below the threshold, the ModSecurity mitigation mechanism was rendered ineffective since the attack could not be detected.

Yeasir et al. (2015) proposed handling requests on behalf of the client using a reverse proxy server that proved to be a good mitigation mechanism against slow header attacks. In their work, the reverse proxy cached the client request until it was complete before handing the completed request off to the web server. However, the reverse proxy server could be a single point of failure unless proper timeouts and thresholds for the connections were determined. Furthermore, determining the appropriate timeout values so as not to disconnect legitimate client traffic was of concern.

Sattar et al. (2016) harnessed the logically centralised nature of SDN to diffuse traffic through a concept referred to as adaptive bubble burst. It mitigated DDoS attacks by establishing the threshold of the serving limit of the web server based on incoming traffic rate measured at the gateway switch. Once the threshold was reached, the adaptive bubble burst mechanism that spread the traffic to multiple replicas of the

target web server was activated. Request completion improved from 4 percent when adaptive bubble burst mechanism was not activated to 81 percent when adaptive bubble burst mechanism was activated. However, the average packet processing time increased from 542μs when adaptive bubble burst was not activated to 776μs per packet when it was activated. Reduction of packet processing overhead was marked for further studies because in the setup used, the controller collected traffic statistics every second.

The study by Hong et al. (2018) explored the use of a controller in SDN to determine whether a client's traffic was legitimate or an attack based on the number of connections with incomplete HTTP requests made within a specified time frame. Before the controller began to track incomplete HTTP requests, the web server had to notify the controller when the maximum number of connections it could respond to was reached. Then, the controller would begin examining subsequent traffic for slow HTTP DDoS attacks.

METHODOLOGY

The use of machine learning algorithms to detect slow HTTP DDoS attacks has proven to be useful and more accurate than other methods. Furthermore, selecting the parameters of the classifier and determining the appropriate features to use in distinguishing legitimate traffic from attack traffic is of importance to improve the results to be obtained (Sahoo et al., 2020). Random Forest has proven to be extremely useful in detecting slow HTTP DDoS attacks over other machine learning detection techniques. However, by using feature and parameter selection techniques, other machine learning methods can be improved upon. In this work, SVM classifier was used to distinguish between legitimate and attack traffic. SVM has two parameters, i.e. regularisation (C) and gamma parameters, which aid in constructing the optimal hyperplane between the data points. Genetic Algorithm was used to select the combination of features that signify the presence of an attack. Moreover, it was employed to tune the SVM parameters to obtain an optimal result.

The dataset used was generated in a GNS3 SDN simulation in which the switch exported NetFlow records obtained to the NetFlow

collector resident on the controller after a timeout value was reached. In the GNS3 simulation software, a Ryu controller was used as the central controller with one openVSwitch as the gateway switch that connected to legitimate HTTP clients and slow HTTP DDoS attackers. Through the NetFlow collector, the controller aggregated all the NetFlow records received. Slow header, POST, and read attacks were simulated using the slowhttptest tool that was selected due to the ease of configuration and tuning of attributes. The attacks were launched from eight computers running on the Ubuntu operating system. Normal traffic was also simulated in addition to legitimate slow clients using python scripts executing on Ubuntu operating systems. The hardware configuration on which GNS3 executed had a processor of Intel(R) Celeron(R) with CPU B820 1.70GHz and a 4GB RAM running on a 64-bit Windows 10 operating system.

Genetic Algorithm

Genetic Algorithm is a meta-heuristic search algorithm based on the natural evolution theory (Agarwal, 2014). Genetic Algorithm begins with an initial set of population consisting of rules generated randomly. Using the concept of survival of the fittest, a new population is created that contains the fittest rules of the current population and their offspring. The fittest rule is usually defined using the classification accuracy of the rules. Rule offspring are formed using crossover and mutation operations. Crossover operation swaps rule pairs to form new pairs of rules. In mutation, randomly selected bits of strings in a rule are inverted. Probabilities for crossover and mutation operations determine the number of times a crossover or mutation operation occurs for chromosomes in a generation. Selecting the rule pair to perform crossover is performed using selection algorithms such as tournament, fitness proportionate, or reward-based selection.

In selecting the features in NetFlow dataset that aided in distinguishing an attack from normal traffic, an initial population size of 10 was defined with crossover probability of 0.5 and gene mutation probability of 0.10. Selection of the chromosomes on which crossover was to be applied was carried out using tournament selection with a size of 3. Generation count of 10 was defined. The fitness function of each chromosome was defined by the accuracy obtained from the SVM classifier. After the features were selected, the regularisation and

gamma parameters for the SVM classifier was tuned using Genetic Algorithm. An initial population size of 10 was defined with crossover probability of 0.5 and gene mutation probability of 0.05. Tournament selection with a tournament size of 3 was applied for selecting chromosomes on which crossover would be carried out. Generation count of 5 was defined. The fitness function of each chromosome was defined by the accuracy of the radial basis function kernel SVM. The range of values defined for the regularisation parameter was from 1 to 10, while gamma values had a range of 0.1 to 1.

Support Vector Machine

The SVM classifier is an algorithm that classifies both linear and nonlinear data by finding the optimal hyperplane and decision boundary that separate the data points in a class from the other (Agarwal, 2014). Kernel functions were used to transform nonlinear data points into a high dimensional space. Use of kernel functions ensured that high accuracy was obtained although the training time may be slow. Radial basis function kernel was used due to some similarities observed between slow HTTP DDoS attacks and slow legitimate traffic. An optimal hyperplane is represented mathematically under the condition of linear separability and linear separable dataset of two points $(x_i, y_i), i = 1,2, \ldots, n$ where $x_i \in \mathbb{R}^n$ and $y_i \in \{+1, -1\}$. Equation 1 represents the correct classification of dataset (Bhati & Rai, 2020; Ye et al., 2019):

$$y_i\big((w.x_i) + b\big) - 1 \geq 0 \; for \; i = 1,2, \ldots\ldots, n \tag{1}$$

where *y* represents two classes that have a binary value, *w* is a weight vector, *x* is an input vector, and *b* is a threshold value. In n-dimensional space, minimising the structural risk of constructing the optimal classification hyperplane is equivalent to solving the constrained optimisation problem with the formula expressed in Equation 2 as:

$$\min[\varphi(w, \varepsilon] = \min\left(\frac{1}{2}\left||w|\right|^2 + C\sum_{i=1}^{N}\varepsilon_i\right) \tag{2}$$
$$s.t.\, y_i(w.x_i + b) \geq 1 - \varepsilon_i, \varepsilon_i \geq 0, i = 1,2, \ldots\ldots, N$$

where the classifier margin is maximised by minimising $\frac{1}{2}\left||w|\right|^2$ and the variables ε_i denotes the extent to which the samples, x_i, violate

the margin. Meanwhile, the regularisation (penalty) parameter C > 0 adjusts the trade-off between minimising the sum of the slack violation errors and maximising the margin (Ma & Guo, 2014)

The optimisation problem in Equation 2 can be expressed through the introduction of Lagrange multiplier, α_i, and the kernel function by the formula in Equation 3 (Liu et al. , 2018):

$$\max[Q(\alpha)] = \max[\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k(x_i, x_j)] \quad (3)$$

$$s.t. \sum_{i=1}^{N} \alpha_i y_i = 0;\ 0 \leq \alpha_i \leq C, i = 1,2, \ldots, N$$

where $k(x_i, x_j)$ is the kernel function. The radial basis function kernel is then represented in Equation 4 (Ma & Guo, 2014):

$$k(x_i, x_j) = \exp\left(-\gamma |x_i - x_j|^2\right) \quad (4)$$

where γ is a constant value to adjust the width of the Gaussian function.

Selective Adaptive Bubble Burst

Selective adaptive bubble burst is a slow HTTP DDoS mitigation technique derived from the synthesis of adaptive bubble burst DDoS mitigation technique in Sattar et al. (2016). In the adaptive bubble burst technique, once an attack is detected, all traffic are spread across all the replica web servers. However, in the selective adaptive bubble burst mitigation technique, the traffic flagged as malicious is rerouted to the replica servers using a method that ensures the monitoring of the flagged traffic's behaviour even when it is rerouted to block the traffic when necessary. The traffic is blocked when it violates the conditions defined relative to the number of replica web servers and the number of times the traffic is rerouted to another web server. The operations of the selective adaptive bubble burst mitigation mechanism can be modelled as a function of the number and IP addresses of the web servers within the network and the connection requests, attack or legitimate, sent to the primary web server as shown in Algorithm 1.

Algorithm 1: Selective Adaptive Bubble Burst Algorithm

Input: SVM Model θ, Netflow record of IP source IP address x is N_{fx}, primary webserver y_1, replica set of webservers $\{y_n\}$ where $\{y_n\} = \{y_2, y_3, \cdots, y_n\} : n \geq 2 \wedge n = |y_1 \cup \{y_n\}|$
Output: Boolean block value μ_x for IP address x
Procedure *onlineSABB*()

```
while network is online do
    Connection i from x to server y is σ_ci(x) ← (x, y)
    μ_x ← false
    Netflow class N_c ← predictClassSVM θ(N_fx)
    if N_c is attack
        Sender-destination pair (x, y) ← getSenderDestPair(N_fx) :
        x ∈ {y_1 ∪ {y_n}}
        if y ≠ y_n
            y' ← serverAfter(y)
            New connections j from x is σ_cj(x) ← (x, y')
        else if y = y_n
            μ_x ← true
        switchPropagate(μ_x)
```

ANALYSIS AND RESULTS

Feature selection was executed first as the necessary features that aided in distinguishing an attack from legitimate traffic were needed for effective choice of SVM parameters. When the features were obtained, the SVM parameter selection phase was launched and the appropriate regularisation and gamma parameters were obtained. The selected features, regularisation parameter, and gamma parameter were then used to perform the classification task using SVM.

Feature Selection and Parameter Tuning Results

A total of 31 features were evaluated for distinguishing an attack from legitimate client traffic. The 31 features comprised 27 NetFlow version 5 features and 4 other features of the time difference between the last and first packets of the flow in seconds, the number of packets per second, bytes per second, and bytes per packet. 13 features were removed because they had the possibility of either causing overfitting of the classifier or had zero values in all tuples for the feature as shown in Table 1. Out of the remaining 18 features, 11 features were selected as features that had a significant effect on distinguishing between an attack and legitimate client traffic as shown in Table 2. Table 3 shows the remaining features that were not selected. Tuning of the SVM

regularisation and gamma parameters was executed using the selected features. The regularisation and gamma values obtained were 8 and 0.798, respectively.

Table 1

Reduced Features in the Dataset

S/N	Feature Name	Description	Rationale for Removal
1	Sys_Uptime	Current time in milliseconds since the export device booted	Possible generalisation
2	Unix_secs	The current count of seconds since 0000 UTC 1970	Possible generalisation
3	Unix_nsecs	Residual nanoseconds since 0000 UTC 1970	Possible generalisation
4	Flow_sequence	Counter of total flow sequence seen	Possible generalisation
5	Sampling Interval	Interval of NetFlow export sampling	Zero value for all tuples
6	Srcaddr	Source IP address	Possible generalization
7	dstaddr	Destination IP address	Possible generalization
8	Nexthop	The IP address of the next-hop router	Zero value for all tuples
9	tos	IP type of service	Zero value for all tuples
10	Src_as	Autonomous system number of the source	Zero value for all tuples
11	Dst_as	Autonomous system number of the destination	Zero value for all tuples
12	Src_mask	Source address prefix mask bits	Zero value for all tuples
13	Dst_mask	Destination address prefix mask bits	Zero value for all tuples

Table 2

Selected Features in the Dataset

S/N	Selected Features	Description
1	Count	Number of flows exported (1-30)
2	Input	Simple Network Management Protocol (SNMP) index of the input interface
3	Output	SNMP index of the output interface
4	dPkts	Packets in the flow
5	dOctets	Total number of layer 3 bytes in the packets of the flow
6	Last	SysUpTime at the time the last packet of the flow was received
7	Diff	The time difference in seconds between the last and first features in the NetFlow version 5 feature set
8	Srcport	TCP/UDP source port number
9	Tcp_flags	Cumulative OR of TCP flags
10	Packets/second	Number of packets per second
11	Bytes/packet	Number of bytes per second

Table 3

Features Rejected by Genetic Algorithm

S/N	Rejected Features	Description
1	Version	The NetFlow export format version number
2	Engine Type	Type of flow switching engine
3	Engine ID	Slot number of the flow switching engine
4	First	Sysuptime at the start of a flow
5	Prot	IP protocol type (TCP = 6; UDP =17)
6	Dst_port	TCP/UDP destination port number
7	Bytes/second	Number of bytes in packets per second

Support Vector Machine Detection Result

The selected features and the SVM parameters obtained were used to define the classification task. The dataset used had 56,891 tuples

in total, which contained 28,446 and 28,445 attack and legitimate client traffic tuples, respectively. A 60:20:20 ratio for training, testing, and validating the model obtained from the classification task was employed. Results obtained are described in Table 4. The Receiver Operating Characteristic (ROC) curve is represented in Figure 1 while the true-positive rate versus the false-negative rate is presented in Figure 2.

Table 4

Classification Result

Performance Metric	Percentage
Accuracy	99.89%
AUC	99.89%
True-positive Rate	99.95%
False-positive Rate	0.18%
False-negative Rate	0.05%

Figure 1

Receiver Operating Characteristic Curve for Slow HTTP DDoS detection

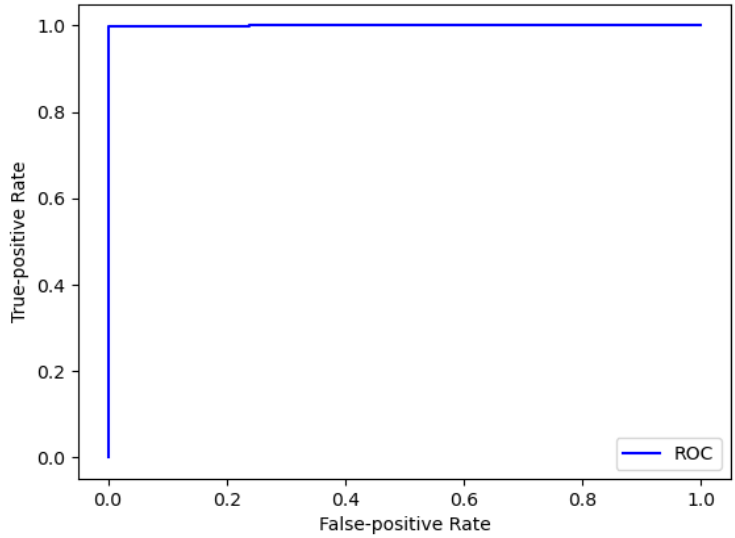
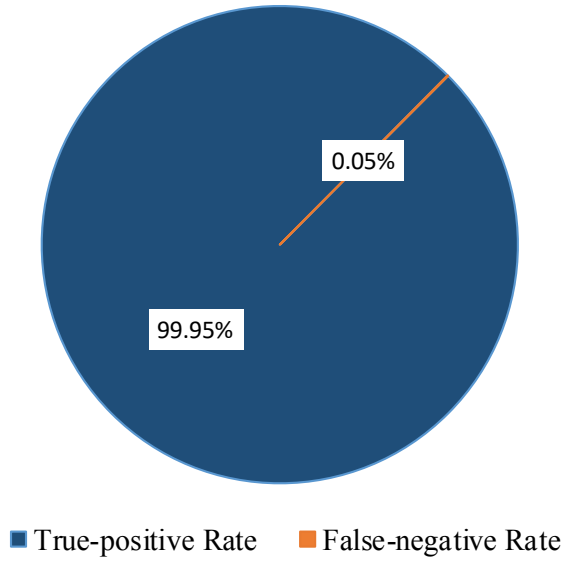


Figure 2

True-positive Rate versus False-negative Rate



From the results obtained, it is established that detection accuracy was high. Furthermore, the true-positive rate showed that most of the attack traffic were classified correctly. The false-positive rate of 0.18 percent indicated the percentage of legitimate traffic, which was misclassified as an attack. Therefore, legitimate traffic had a significantly low chance of being tagged as attack traffic. The low false-positive rate and the high true-positive rate implied that most slow HTTP DDoS attack traffic were detected while a low amount of legitimate traffic was misclassified as malicious. Although the false-positive rate would cause some connection issues on the legitimate client side, this was negligible as compared to the true-positive rate, which kept the network free from attack traffic. Similarly, the low false-negative rate of 0.05 percent implied that a negligible amount of attack traffic was misclassified as legitimate. Although some slow HTTP DDoS attack traffic would arrive at the target web server due to the false-negative rate obtained, its effect would be negligible since the high true-positive rate had reduced a huge chunk of the attack traffic.

DISCUSSION

The results obtained by the Genetic Algorithm with SVM classifier buttressed the use of machine learning for detecting anomalies. Furthermore, it showed that feature selection and classifier parameter tuning enhanced the quality of results obtained when the accuracy of 99.89 percent obtained in this work was juxtaposed with the accuracy of 93.10 percent obtained in Perez-Diaz et al.'s (2020) work. Although the methodology and dataset used by Perez-Diaz et al. (2020) differed from this work, it is significant to note that the use of NetFlow instead of FPC by applying feature selection and parameter tuning enhanced the detection ability of SVM. This means that the slow HTTP DDoS detection ability of SVM was as good as that of Random Forest as seen in the literature though it depended on the proper tuning of SVM parameters for effective classification.

CONCLUSION

In this paper, the detection of slow HTTP DDoS attacks using Genetic Algorithm to select appropriate NetFlow features and tune SVM parameters was executed. The proposed methodology applied radial basis function kernel-based SVM to classify the instances in the dataset into the attack and benign traffic. Since accurate detection of the attack needs a proper mitigation technique, a novel technique called selective adaptive bubble burst was presented although yet to be implemented. The use of slow HTTP DDoS to circumvent volumetric DDoS mitigation mechanisms and the low amount of research into detecting the three major slow HTTP DDoS attacks influenced the execution of this research. Furthermore, the absence of a publicly available slow HTTP DDoS dataset based on NetFlow version 5 necessitated the creation of a dataset by simulating an SDN network in GNS3. Detection of slow HTTP DDoS attacks was executed using Genetic Algorithm to select NetFlow version 5 features and tune the regularisation and gamma parameters of the SVM classifier used. Detecting the slow HTTP DDoS attack effectively enables the mitigation mechanism to be triggered and applied to the traffic that caused the event. The algorithm for the selective adaptive bubble burst mitigation process was also proposed. This paper contributes towards several existing research on slow HTTP DDoS attacks and has

shown that feature selection and tuning of SVM parameters enhance detection of slow HTTP DDoS attacks. Furthermore, it contributes to ongoing research on the use of meta-heuristic algorithms such as Genetic Algorithm and PSO to enhance machine learning detection of a variety of network attacks. Moreover, a new approach to mitigate slow HTTP DDoS attacks was presented.

An area of future research is to implement the selective adaptive bubble burst DDoS mitigation algorithm developed in an SDN controller to evaluate its effectiveness. This implies that the detection mechanism needs to be operational in the network by uploading the detection module to the controller. Furthermore, a machine learning framework that learns from the traffic classified in a live or simulated network is another possible area for further studies. This enables the detection mechanism to be updated based on the results it obtains after classifying live traffic and ensures that the detection mechanism remains relevant without a need to create an entirely new model at intervals.

ACKNOWLEDGMENT

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

REFERENCES

Agarwal, S. (2014). Data mining: Data mining concepts and techniques. *Proceedings - 2013 International Conference on Machine Intelligence Research and Advancement, ICMIRA 2013*, 203–207. https://doi.org/10.1109/ICMIRA.2013.45

Ameyed, D., Jaafar, F., & Fattahi, J. (2015). A slow read attack using cloud. *Proceedings of the 2015 7th International Conference on Electronics, Computers and Artificial Intelligence, ECAI 2015*, SSS33–SSS38. https://doi.org/10.1109/ECAI.2015.7301202

Beigi-Mohammadi, N., Barna, C., Shtern, M., Khazaei, H., & Litoiu, M. (2017). CAAMP: Completely automated DDoS attack mitigation platform in hybrid clouds. *2016 12th International Conference on Network and Service Management, CNSM 2016*

and Workshops, 3rd International Workshop on Management of SDN and NFV, ManSDN/NFV 2016, and International Workshop on Green ICT and Smart Networking, GISN 2016, 136–143. https://doi.org/10.1109/CNSM.2016.7818409

Benzekki, K., El Fergougui, A., & Elbelrhiti Elalaoui, A. (2016). Software-defined networking (SDN): A survey. *Security and Communication Networks*, *9*(18), 5803–5833. https://doi.org/10.1002/sec.1737

Bhunia, S. S., & Gurusamy, M. (2017). Dynamic attack mitigation using SDN. *2017 27th International Telecommunication Networks and Applications Conference, ITNAC 2017*, *2017-Janua*, 1–6. https://doi.org/10.1109/ATNAC.2017.8215430

Brynielsson, J., & Sharma, R. (2015). Detectability of low-rate HTTP server DoS attacks using spectral analysis. *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2015*, 954–961. https://doi.org/10.1145/2808797.2808810

Calvert, C. L., & Khoshgoftaar, T. M. (2019). Impact of class distribution on the detection of slow HTTP DoS attacks using Big Data. *Journal of Big Data*. https://doi.org/10.1186/s40537-019-0230-3

Cambiaso, E., Papaleo, G., & Aiello, M. (2017). Slowcomm: Design, development and performance evaluation of a new slow DoS attack. *Journal of Information Security and Applications*, *35*, 23–31. https://doi.org/10.1016/j.jisa.2017.05.005

Cambiaso, E., Papaleo, G., Chiola, G., & Aiello, M. (2013). Slow DoS attacks: Definition and categorisation. *International Journal of Trust Management in Computing and Communications*, *1*(3/4), 300. https://doi.org/10.1504/ijtmcc.2013.056440

Cusack, B., & Tian, Z. (2016). Detecting and tracing slow attacks on mobile phone user service. *Proceedings of the 14th Australian Digital Forensics Conference, ADF 2016*, (December), 4–10. https://doi.org/10.4225/75/58a54b013185a

Dabbagh, M., Hamdaoui, B., Guizani, M., & Rayes, A. (2015). Software-defined networking security: pros and cons. *IEEE Communications Magazine*, *53*(6), 73-79. https://doi.org/10.1109/MCOM.2015.7120048.

Dantas, Y. G., Fonseca, I. E., & Nigam, V. (2017). Slow TCAM exhaustion DDoS attack. In *IFIP International Conference on ICT Systems Security and Privacy Protection* (pp. 17–31). https://doi.org/10.1007/978-3-319-58469-0

Dhanapal, A., & Nithyanandam, P. (2019). The slow HTTP distributed denial of service attack detection in cloud. *Scalable Computing*, *20*(2), 285–298. https://doi.org/10.12694/scpe.v20i2.1501

Ezekiel, S., Divakaran, D. M., & Gurusamy, M. (2017). Dynamic attack mitigation using SDN. *2017 27th International Telecommunication Networks and Applications Conference, ITNAC 2017*, *2017-Janua*, 1–6. https://doi.org/10.1109/ATNAC.2017.8215430

Foñseca, I. E., & Nigam, V. (2016). *Mitigating high-rate application layer DDoS attacks in software defined networks*.

Hamad, D. J., Yalda, K. G., & Okumuş, I. T. (2016). Getting traffic statistics from network devices in an SDN environment using OpenFlow. *Information Technology and Systems 2015*, (April), 951–956.

Hong, K., Kim, Y., Choi, H., & Park, J. (2018). SDN-assisted slow HTTP DDoS attack defense method. *IEEE Communications Letters*, *22*(4), 688–691. https://doi.org/10.1109/LCOMM.2017.2766636

Idhammad, M., Afdel, K., & Belouch, M. (2018). Detection System of HTTP DDoS attacks in a cloud environment based on information theoretic entropy and random forest. *Security and Communication Networks*, *2018*. https://doi.org/10.1155/2018/1263123

Jaafar, G. A., Abdullah, S. M., & Ismail, S. (2019). Review of recent detection methods for HTTP DDoS attack. *Journal of Computer Networks and Communications*, Vol. 2019. https://doi.org/10.1155/2019/1283472

Jazi, H. H., Gonzalez, H., Stakhanova, N., & Ghorbani, A. A. (2017). Detecting HTTP-based application layer DoS attacks on web servers in the presence of sampling. *Computer Networks*, *121*, 25–36. https://doi.org/10.1016/j.comnet.2017.03.018

Kemp, C., Calvert, C., & Khoshgoftaar, T. M. (2018). Utilizing netflow data to detect slow read attacks. *Proceedings - 2018 IEEE 19th International Conference on Information Reuse and Integration for Data Science, IRI 2018*, 108–116. https://doi.org/10.1109/IRI.2018.00023

Latah, M., & Toker, L. (2018). Artificial intelligence enabled software-defined networking: a comprehensive overview. *IET networks*, *8*(2), 79-99. https://doi.org/10.1049/iet-net.2018.5082

Liu, H., & Kim, M. S. (2010). Real-time detection of stealthy DDoS attacks using time-series decomposition. *IEEE International Conference on Communications*. https://doi.org/10.1109/ICC.2010.5501975

Liu, S., Wang, L., Qin, J., Guo, Y., & Zuo, H. (2018). An intrusion detection model based on IPSO-SVM algorithm in wireless sensor network. *Journal of Internet Technology*, *19*(7), 2125–2134. https://doi.org/10.3966/160792642018121907015

Lukaseder, T., Maile, L., Erb, B., & Kargl, F. (2018). SDN-assisted network-based mitigation of slow DDoS attacks. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, *255*, 102–121. https://doi.org/10.1007/978-3-030-01704-0_6

Ma, Y., & Guo, G. (2014). Support vector machines applications. In *Support Vector Machines Applications* (Vol. 9783319023). https://doi.org/10.1007/978-3-319-02300-7

Muraleedharan, N., & Janet, B. (2018). Behaviour analysis of HTTP based slow denial of service attack. *Proceedings of the 2017 International Conference on Wireless Communications, Signal Processing and Networking, WiSPNET 2017*, *2018-Janua*, 1851–1856. https://doi.org/10.1109/WiSPNET.2017.8300082

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

Park, J. (2015). Analysis of slow read DoS attack and countermeasures on web servers. *International Journal of Cyber-Security and Digital Forensics*, *4*(2), 339–353. https://doi.org/10.17781/p001550

Perez-Diaz, J. A., Valdovinos, I. A., Choo, K. K. R., & Zhu, D. (2020). A flexible SDN-based architecture for identifying and mitigating low-rate DDoS attacks using machine learning. *IEEE Access*, *8*, 155859–155872. https://doi.org/10.1109/ACCESS.2020.3019330

Polat, H., & Polat, O. (2021). An intelligent software defined networking controller component to detect and mitigate denial of service attacks. *Journal of Information and Communication Technology*, *20*(1), 57–81. https://doi.org/10.32890/jict.20.1.2021.6288

Sahoo, K. S., Tripathy, B. K., Naik, K., Ramasubbareddy, S., Balusamy, B., Khari, M., & Burgos, D. (2020). An evolutionary SVM model for DDOS attack detection in software defined networks. *IEEE Access*, *8*, 132502–132513. https://doi.org/10.1109/ACCESS.2020.3009733

Sattar, D., Matrawy, A., & Adeojo, O. (2016). Adaptive bubble burst (ABB): Mitigating DDoS attacks in software-defined networks. *2016 17th International Telecommunications Network Strategy and Planning Symposium, Networks 2016 - Conference Proceedings*, 50–55. https://doi.org/10.1109/NETWKS.2016.7751152

Schehlmann, L., & Baier, H. (2013). COFFEE : A concept based on OpenFlow to filter and erase events of botnet activity at high-speed nodes. *GI-Jahrestagung*, 2225–2239.

Shafieian, S., Zulkernine, M., & Haque, A. (2015). CloudZombie: Launching and detecting slow-read distributed denial of service attacks from the Cloud. *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*, 1733–1740. https://doi.org/10.1109/CIT/IUCC/DASC/PICOM.2015.261

Shtern, M., Sandel, R., Litoiu, M., Bachalo, C., & Theodorou, V. (2014). Towards mitigation of low and slow application DDoS attacks. *Proceedings - 2014 IEEE International Conference on Cloud Engineering, IC2E 2014*, (Vm), 604–609. https://doi.org/10.1109/IC2E.2014.38

Bhati, B. S., & Rai, C. S. (2020). Analysis of support vector machine-based intrusion detection techniques. *Arabian Journal for Science and Engineering*, *45*(4), 2371-2383. https://doi.org/10.1007/s13369-019-03970-z

Singh, K. J., & De, T. (2015). An approach of ddos attack detection using classifiers. In *Emerging Research in Computing, Information, Communication and Applications* (pp. 429-437). Springer, New Delhi. https://doi.org/10.1007/978-81-322-2550-8

Siracusano, M., Shiaeles, S., & Ghita, B. (2018, October). Detection of lddos attacks based on tcp connection parameters. In *2018 Global Information Infrastructure and Networking Symposium (GIIS)* (pp. 1-6). IEEE. https://doi.org/ 10.1109/GIIS.2018.8635701

Suroto, S. (2017). A review of defense against slow HTTP attack. *JOIV : International Journal on Informatics Visualization*, *1*(4), 127. https://doi.org/10.30630/joiv.1.4.51

Swami, R., Dave, M., & Ranga, V. (2019a). Defending DDoS against software defined networks using entropy. *Proceedings - 2019 4th*

International Conference on Internet of Things: Smart Innovation and Usages, IoT-SIU 2019, 1–5. https://doi.org/10.1109/IoT-SIU.2019.8777688

Swami, R., Dave, M., & Ranga, V. (2019b). Software-defined networking-based DDoS defense mechanisms. *ACM Computing Survey*, *52*(2), 36. https://doi.org/10.1016/B978-0-12-375000-6.00124-5

Tayama, S., & Tanaka, H. (2017, June). Analysis of slow read DoS attack and communication environment. In *International Conference on Mobile and Wireless Technology* (pp. 350-359). Springer, Singapore. https://doi.org/10.1007/978-981-10-5281-1

Tripathi, N., & Hubballi, N. (2018). Slow rate denial of service attacks against HTTP/2 and detection. *Computers and Security*, *72*, 255–272. https://doi.org/10.1016/j.cose.2017.09.009

Tripathi, N., Hubballi, N., & Singh, Y. (2016). How secure are web servers? An empirical study of slow HTTP DoS attacks and detection. *Proceedings - 2016 11th International Conference on Availability, Reliability and Security, ARES 2016*, 454–463. https://doi.org/10.1109/ARES.2016.20

Ye, Z., Sun, Y., Sun, S., Zhan, S., Yu, H., & Yao, Q. (2019). Research on network intrusion detection based on support vector machine optimized with grasshopper optimization algorithm. *2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, *1*(41301371), 378–383. https://doi.org/10.1109/IDAACS.2019.8924234

Yeasir, M., Morshed, M., & Fakrul, M. (2015). A practical approach and mitigation techniques on application layer DDoS attack in web server. *International Journal of Computer Applications*, *131*(1), 13–20. https://doi.org/10.5120/ijca2015907209

Yuan, B., Zou, D., Jin, H., Yu, S., & Yang, L. T. (2020). HostWatcher: Protecting hosts in cloud data centers through software-defined networking. *Future Generation Computer Systems*, *105*, 964-972. https://doi.org/10.1016/j.future.2017.04.023

Zolotukhin, M., Hamalainen, T., Kokkonen, T., & Siltanen, J. (2016). Increasing web service availability by detecting application-layer DDoS attacks in encrypted traffic. *2016 23rd International Conference on Telecommunications, ICT 2016*. https://doi.org/10.1109/ICT.2016.7500408