7

DDoSDetect: A Behavioral Detection System for HTTP GET Flood Attacks

[1]Tinubu C.O., [2]Falana O.J., [3]Aborisade D.O., [4]Adejimi O.A and [5]Akinmusire C.B*

[1,2,3,4,5]Department of Computer Science, Federal University of Agriculture, Abeokuta, Nigeria.

tinubuco@funaab.edu.ng[1]*, falanaoj@funaab.edu.ng[2], aborisadedo@funaab.edu.ng[3], adejimioa@funaab.edu.ng[4], akinmusirecharles@gmail.com[5]

ABSTRACT

Application layer Distributed Denial of Service (DDoS) attacks are becoming increasingly prevalent on Internet based web services. Severe HTTP GET flood attacks are often launched by attackers, wherein web servers become overwhelmed with superfluous GET requests. This exhausts the processing and connectivity resources of the servers, hence resulting in service failures. As availability of services is key to online service providers, this work presents a behavioral based detection system named 'DDoSDetect' for HTTP GET flood attacks. In DDoSDetect, a traffic monitor tracks incoming GET requests sent to the server. Four distinct behavioral features such as the rate of requests, time interval between successive requests, session rate and the frequency of requests on a web page, are employed to analyze user's behaviors. A detection unit driven by three phases: LRP (Legitimate Request Phase), IRP (Illegitimate Request Phase) and SVM-RC (Support Vector Machine-Request Classifier) is designed to intelligently classify incoming GET requests. Illegitimate GET requests are dropped while legitimate requests are forwarded to the web server. DDoSDetect is implemented and results obtained from evaluation processes signify the effectiveness of the system towards the mitigation of HTTP GET flood attacks. Also, the system is real time and capable of handling sophisticated attacks.

Keywords - Application layer, availability, DDoS attacks, HTTP GET flood, Internet, web server.

1. Introduction

Modern industries and organizations depend on the Internet to provide web-based services to their customers/clients. Web applications provide information and services to users without time or space constraints (Praseed and Thilagam, 2018). As web applications are becoming increasingly relevant for businesses, financial institutions and government, a growing number of online services utilize application layer protocols such as Hypertext Transfer Protocol (HTTP), Domain Name System (DNS), Simple Mail Transfer Protocol (SMTP), etc. However, these applications have become targets to many distributed and sophisticated attacks. Such attacks include Phishing attacks, Password attacks, Malware attacks, Man-in-the-middle attacks and Distributed Denial of Service attacks. These attacks are being launched to compromise the confidentiality, integrity and availability of targeted systems. It therefore becomes pertinent to ensure the security of web servers at all times as web applications are greatly influencing businesses and services.

Distributed Denial of Service (DDoS) attacks are one of the most devastating attacks on the Internet. A DDoS attack is a coordinated attack on the availability of services of a given target system, being launched through many compromised computing systems (Singh and Gupta, 2016). DDoS attacks are aimed at exhausting the processing and connectivity resources of a victim server, resulting in a partial or total unavailability (Yusof *et al.,* 2019). They are deliberate attempts by attackers to degrade the quality of service of a system. Usually, the

attackers utilize a network of compromised Internet-connected devices known as a botnet to flood targeted servers. Flooding-based DDoS attacks seek to deny services to legitimate users by invoking vast amounts of bogus requests, thereby exhausting key resources of the victim (Bhardwaj *et al*., 2016). These attacks cause devastating effects such as service interruption, reduced quality of service, loss of confidentiality, reputational damages, huge financial losses, breach of contracts and loss of users' trust.

An emerging form of Flooding DDoS attacks is persistently targeting the application layer. In recent years, cybercriminals are now turning to Application layer DDoS attacks. Unlike Network layer DDoS attacks which exhausts the network bandwidth of a server, Application layer attacks target the exhaustion of resources of a web server such as CPU cycles, database cycles, memory and/or socket connections. Network layers attacks are volumetric attacks relying on a large chunk of network layer packets to deplete the bandwidth. Network layer protocols such as UDP (User Datagram Protocol) and ICMP (Internet Control Message Protocol) are often employed to launch such attacks.

In Application layer DDoS attacks, the application layer of the Internet is exploited to disrupt the normal flow of traffic to a web server. Multiple requests from a compromised host who masquerades as legitimate users flood the server and exhaust its resources. Usually, attackers target high-profile web servers that provide specific services. Application layer attacks can pull down a server much faster and with higher stealth than network layer attacks (Praseed and Thilagam, 2018). With application layer attacks, users lose interests in the services of the victim organization if periodic attacks leave the website(s) inaccessible.

Being the dominant part of the Internet, the Hypertext Transfer Protocol (HTTP) is highly susceptible to flooding attacks (Singh *et al.,* 2017). HTTP is a client-server protocol and the foundation of any data exchange on the web. In an HTTP flood attack, the attacker exploits seemingly-legitimate HTTP GET or POST requests to overwhelm a targeted server. As these requests have legitimate IP addresses and are sent through valid TCP connections, they become difficult to detect with existing defense mechanism such as Intrusion Detection System (IDS), filtering or blacklisting techniques.

HTTP GET Flood attack is the most common DDOS attack on the application layer of a network (Mirvaziri, 2017). During the attack, a bot sends multiples of GET requests to the server and awaits responses like a legitimate user. Consequently, the request queue increases spontaneously and the server becomes obliged to allocate the maximum resources

possible in response to every GET request (Jaafar *et al.,* 2019). The computational resources of the web server gets exhausted as the server processes and responds to malicious requests, thereby denying subsequent incoming requests from legitimate users. Detecting HTTP GET flood attacks is particularly challenging as malicious request packets often appear similarly to normal request packets (Najafabadi *et al.,* 2017). In an attempt to evade detection, a stealthy attacker launches sophisticated attacks that closely mimic the behaviors of genuine GET requests.

Our Contribution

The rising sophistication of HTTP GET flood attacks necessitates continuous research efforts towards preserving the availability of web services. In this work, we develop an efficient behavioral-based detection system namely DDoSDetect for the mitigation of HTTP GET flood attacks. The system is based on four carefully selected behavioral features that can distinctly differentiate genuine GET requests from the illegitimate requests. More precisely, the system is an intelligent one targeted at sophisticated attacks. Moreover, the system is real-time and ensures an overall availability of web servers.

The rest of the paper is organized as follows. Section 2 discusses the existing works on the detection of HTTP GET flood attacks. Section 3 describes the methodology of the proposed system 'DDoSDetect', giving details on its feature selection, architecture and algorithms. In Section 4, the obtainable experimental results are presented and evaluated. Finally, in Section 5, we conclude the work.

2. Literature Review

Several efforts in literatures have focused on the detection of HTTP GET flood attacks. Some studies focused on puzzle-based methods (Ko *et al.,* 2010), load balancing mechanism (Spagna *et al.,* 2013), hardware mechanism (Jin *et al.,* 2010), access control lists and hidden cache of the browser (Ak *et al.,* 2012) and IP Hopping (Choi *et al.,* 2010). But none of these approaches proffer effective solutions for Application-layer DDoS attacks detection.

Some efforts have been geared towards analyzing user's biometric behavior for the detection of malicious traffic. Such approaches involve leveraging on features based on user's interactions with the system for their authentication. The use of keystrokes, mouse dynamics and interaction with a graphical user interface (GUI) (Stevanovic and Vlajic, 2014; Jin *et al.,* 2010; Abramson and Aha, 2013; Shen *et al.,* 2013; Bravo and Mauricio, 2018) have been considered. Nevertheless, real users

and robots cannot be differentiated while using these approaches.

Detecting GET flood attacks by monitoring anomalies in user's browsing behaviors have been greatly explored. Characteristics such as speed of browsing pages (Yatagai *et al.,* 2007), number of users and requests (Ranjan *et al.,* 2008), entropy of request type (Huang *et al.,* 2014), request count (Singh *et al.,* 2017), click number of web objects (Zhou *et al.,* 2014; Wang *et al.,* 2014), IP address (Campo *et al.,* 2013), number of bytes sent in 1 second (Zolotukhin *et al.,* 2016) users browsing process (Dick and Scheffer, 2016), web page requested (Saravanan *et al.,* 2016), attack rate (Thapngam *et al.,* 2011) etc., have been employed to detect variations from normal behaviors. Many of these models have germane weaknesses as their choices of features are not sufficient to counter sophisticated attacks.

In (Xie and Yu, 2009), the authors proposed modeling the sequence order of legitimate page requests and characterizing legitimate and suspicious browsing behaviors based on Hidden semi Markov Model (HsMM). Liao *et al.* (2015) adopted Support Vector Machine (SVM) to analyze user's browsing behaviors based on access frequencies, especially request time interval and frequency of requests. Miu *et al.* (2016) proposed monitoring the sequence of user's web page access to identify anomalous users. Singh *et al.* (2018) also proposed user-behavioral analytics for detection of HTTP GET flood attacks. These approaches while detecting attacks could not attain efficiency as they are not updated in real time and cannot handle sophisticated attacks.

These existing researches indicate that the variations of behavioral patterns between legitimate users and attackers have great potentials to distinguish human users from attack bots. Hence, this paper presents an improved behavioral-based system for the detection of HTTP GET flood attacks.

3. Methodology

There is an inherent need to improve detection solutions for HTTP GET flood attacks through intelligent and real-time mechanisms. The detection of these attacks requires a high-level monitoring of all GET requests sent to the server. The proposed system named 'DDoSDetect' is built on a behavioral analysis of incoming HTTP GET requests.

Four features are employed to analyze the behaviors of all GET requests sent to the server.

These features are carefully selected and targeted at distinguishing legitimate users from sophisticated bots. They are:

- Rate of GET requests: this is the number of GET requests sent to the server in a specific time interval, measured in *requests per second*. This feature is adopted by

considering that the request rate of bots will be higher than those of legitimate users.

- Time interval between successive GET requests: this is the time difference between two successive GET requests sent from a particular IP address, *measured in seconds.*
- Session rate: this is the number of sessions initiated by a particular IP address during a specific period of time, measured in *sessions per seconds.*
- Frequency of requests: this is the number of requests made on a particular web page in one session.

Normal traffic from a dataset was investigated to obtain typical browsing behaviors of users over a period of time. This observation runs on a time frame of 180 minutes. The behaviors of the legitimate GET requests are used as standards to flag anomalies in the proposed system. Such legitimate behaviors observed within each time frame are analyzed, and the following characteristics are deduced.

R_L = *maximum rate of GET requests from a legitimate user*

T_L = *minimum time interval between successive legitimate requests*

S_L = *maximum session rate of a legitimate user*

N_L = *maximum number of legitimate requests on a web page in a session*

A function F is derived from the characteristics obtained from legitimate GET requests to compute two classification zones, namely Safe and Danger zones. The Safe zone (Sz) refers to a class in which the resources of a server are safe from being overwhelmed. The Danger zone (Dz) refers to a class in which the resources of a server are at a great risk of being overwhelmed, resulting to a DDoS attack.

The function F is computed as represented in (1).

$$F = \sum((R_L), (T_L), (S_L), (N_L)) \qquad (1)$$

For each connected user, Index I using (2) is calculated from the behavioral features of the incoming GET requests.

$$I = 1 + \frac{(R_i + N_i)(S_i)}{T_i} \qquad (2)$$

where R_i = rate of requests, N_i = frequency of requests, S_i = session rate and T_i = time interval.

Safe zone: A computed Index I is in a Safe zone Sz if:

$$N_i < I \leq F \qquad (3)$$

Danger zone: A computed Index I is in a Danger zone Dz if:

$$R_i \leq I > F \qquad (4)$$

The Detection System

The detection system 'DDoSDetect' essentially is comprised of two components namely: Traffic Monitor and Detection Unit as shown in Figure. 1. The traffic monitor is an interface showing the details of each HTTP GET request sent to the web server. The traffic monitor tracks the flow of all requests to the web server. Through the monitor, the behavioral features of incoming GET requests are captured for analysis. The detection unit is responsible for the classification of GET requests as either legitimate or illegitimate requests.

The Detection unit of DDoSDetect is divided into three phases. These are:

- **The Legitimate Request Phase (LRP):** This phase is designed to verify if the incoming GET requests fall within the Safe zone. The verification is done through the computed Index of the user's behavioral features. Requests that are classified as being legitimate are forwarded to the web server. Otherwise, the requests are passed on to the Illegitimate Request Phase.
- Illegitimate Request Phase (IRP): This phase is designed to verify if the incoming GET requests fall within the Danger zone. The verification, which is achieved through the computed Index of the user's behavioral features, determines if the GET requests are illegitimate requests. Such requests are blocked, thus preventing the web server from committing resources to its processing.
- **Support Vector Machine-Request Classifier (SVM-RC):** Some attacks are highly sophisticated and carefully designed by rational attackers such as to evade detection. In DDoSDetect, considerations have been made for such attacks as they may not fall within the classified Safe and Danger zones. Consequently, an intelligent phase called the Support Vector Machine-Request Classifier (SVM-RC) is designed to detect such malicious requests. When the Legitimate Request Phase (LRP) and Illegitimate Request Phase (ILP) cannot classify a request, the trained Support Vector Machine (SVM) classifier is applied to identify the attacks.

The method, as shown in Algorithm 1 is based on the Support Vector Machine (SVM), a machine learning classifier. A SVM algorithm builds a robust classification model. Support Vector Machines operates on the principle of creating a hyperplane that separates dataset into classes. The extreme vectors that help in creating the hyperplane are chosen. The dimension of the hyperplane is dependent on the features present in the data set. As there are four proposed features, the hyperplane will be a 3-dimension plane.

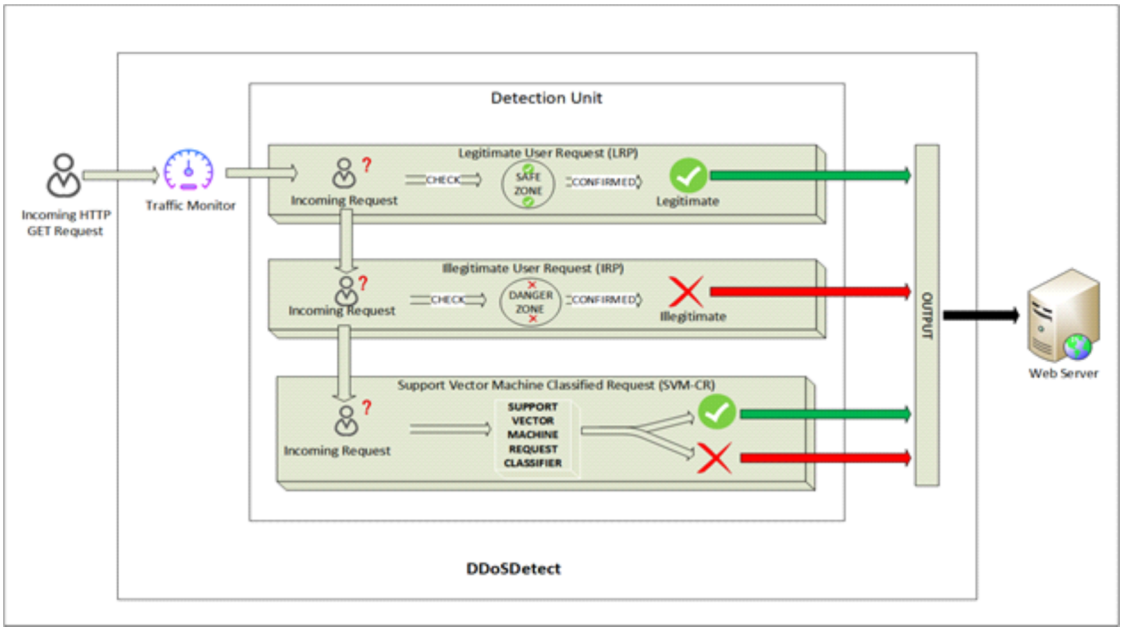

Fig. 1: Architecture of the Proposed 'DDoSDetect'

Algorithm 1: Support Vector Machine-Request Classifier (SVM-RC)

Input: request_rate(), time_interval(), session_rate(), frequency_request(), train_data, test_data
Output: Legitimate / Illegitimate

```
1. Train ← read.csv (file.choose(train_data))
2. Test ← read.csv (file.choose(test_data))
3. model ← SVM (Target~class, data=Train, kernel= 'linear')  //Train model
4. testing ← predict (model, Test)  //Test model
5. result ← predict (model, [request_rate, time_interval, session_rate, fequency_request])
6. if result == 1
      return Legitimate
7. else
      return Illegitimate
```

4. IMPLEMENTATION AND EVALUATION OF RESULTS

The system implementation of 'DDoSDetect' was done in JetBrains PyCharm Integrated Development Environment (IDE) using PCs with 64-bit Windows Operating System, Intel core i3, 1 Terabyte Hard disk drive, 4GB RAM and 2.00GHz CPU. Low Orbit Ion Cannon (LOIC), a software tool was employed to launch a DDoS attack on the web server.

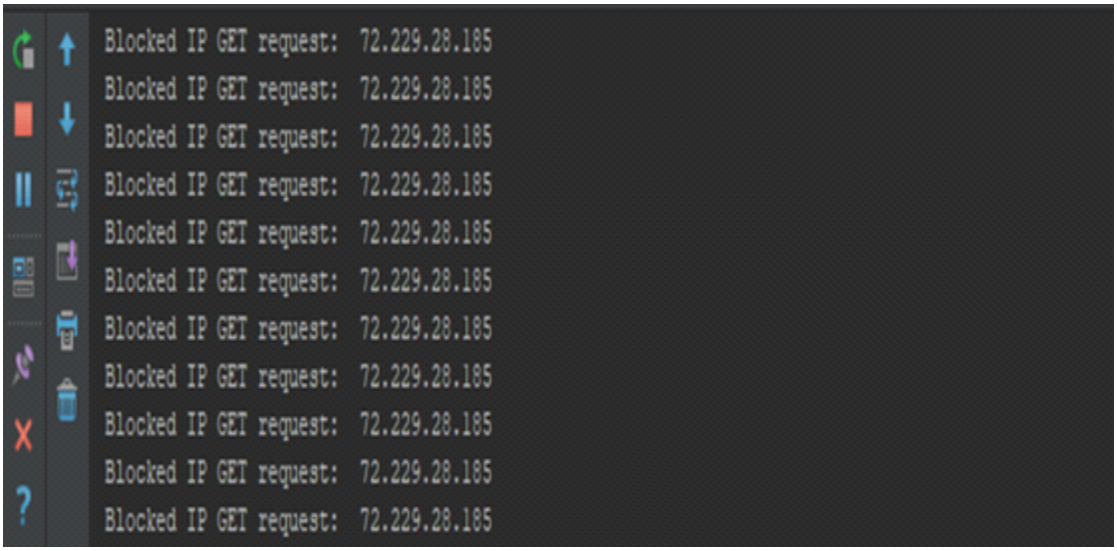

Fig. 2: Output from DDoSDetect showing a malicious GET request

Figure 2 shows the reaction of DDoSDetect on detecting a malicious HTTP GET request. The performance of DDoSDetect is evaluated using metrics such as: Accuracy, Precision, Recall and F-measure.

i) Accuracy: is the most intuitive performance measure. It reflects the overall effectiveness of a classifier.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

ii) Precision: is the ratio of correctly predicted observations to the total predicted positive observations.

$$\text{Precision} = \frac{TP}{TP + FP}$$

iii) Recall: is the ratio of correctly predicted positive observations to the total observations in actual class.

$$\text{Recall} = \frac{TP}{TP + FN}$$

iv) F-measure: is the weighted mean of the precision and recall of the classifier.

$$\text{F-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

where,

True Positive (TP) denotes the number of malicious GET requests correctly classified.

False Negative (FN) denotes the number of malicious GET requests erroneously classified as benign requests.

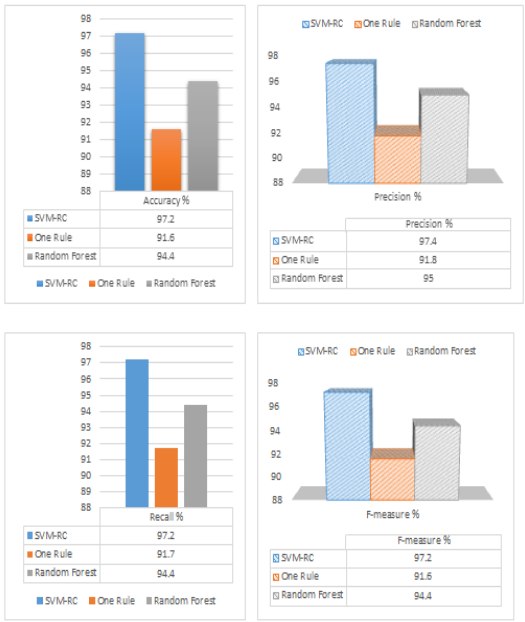

Fig. 3: Performance Comparison of SVM-RC in DDoSDetect with One Rule and Random Forest classifiers

False Positive (FP) denotes the number of benign GET requests erroneously classified as malicious requests.
True Negative (TN) denotes the number of benign GET requests correctly classified.

Performance Comparison of DDoSDetect with Traditional Models

To compare the effectiveness of DDoSDetect with traditional models, several experiments were performed using different classifiers in a JetBrains PyCharm Integrated Development Environment (IDE). The Support Vector Machine-Request Classifier (SVM-RC) in DDoSDetect, OneRule and Random Forest algorithms were used separately on the test dataset based on the four selected features to investigate the performance of the models. The results of the experiments as shown in Figure 3 show that the Support Vector Machine-Request Classifier (SVM-RC) has the best accuracy of 97.2% in comparison with other classifiers as OneRule and Random Forest algorithms with an accuracy of 91.6% and 94.4% respectively.

4. CONCLUSION

Web servers that run on Hypertext Transfer Protocol (HTTP) are vulnerable to Distributed Denial of Service (DDoS) attacks. HTTP GET Flood attack is the most common DDoS attack on the application layer of a network. Application layer attacks target the exhaustion of resources of a web server such as CPU cycles, database cycles, memory or socket connections. These attacks cause unavailability of services as attackers utilize botnets to send huge numbers of malicious GET requests to overwhelm the web server. The resulting effects include service interruptions, reduced quality of service, reputational damages, huge financial losses, breach of contracts and loss of users' trust. In this work, a behavioral based detection system was presented for HTTP GET flood attacks. The system named 'DDoSDetect' is based on four behavioral features. DDoSDetect

leverages on the observed behaviors of legitimate GET requests to detect anomalies in the behaviors of attacking bots to distinguish legitimate requests from malicious requests. Incoming GET requests are classified by DDoSDetect such that malicious requests are dropped and the legitimate requests are forwarded to the web server. Experimental results show that the detection accuracy of DDoSDetect outperforms other classifiers, and thus effective at mitigating HTTP GET flood attacks.

References

Abramson, M., & Aha, D. (2103). User authentication from web browsing behavior. 268-273.

Ak, M., George, L., Govind, K., & Selvakumar, S. (2012). Threshold Based Kernel Level HTTP Filter (TBHF) for DDoS mitigation. International Journal of Computer Network and Information Security, 4(31).

Bhardwaj, A., Subrahmanyam, G., Avasthi, V., Sastry, H., & Goundar, S. (2016). DDoS Attacks, New DDoS Taxonomy and Mitigation Solutions- A Survey. 793-798.

Bravo, S., & Mauricio, D. (2018). Distributed Denial of Service Attack Detection in Application Layer Based on User Behavior. Webology, 15.

Campo, G., Cristina, C., de Diego, I., & Enrique, C. (2013). Detecting denial of service by modelling web-server behavior. Computers and Electrical Engineering, 39, 2252-2262.

Choi, Y., Oh, J., Jang, J., & Ryou, J. (2010). Integrated DDoS attack defense infrastructure for effective attack prevention. 1-6.

Dick, U., & Scheffer, T. (2016). Learning to control a structured-prediction decoder for detection of HTTP-layer DDoS attackers. Machine Learning, 104, 385-410.

Huang, C., Wang, J., Wu , G., & Chen, J. (2014). Mining Web User Behaviors to Detect Application Layer DDoS Attacks. JSW, 9, 985-990.

Jaafar, G., Abdullah, S., & Ismail, S. (2019). Review of recent detection methods for HTTP DDoS attack. Journal of Computer Networks and Communications.

Jin, J., Nodir, N., Im, C., & Nam, S. (2010). Mitigating HTTP GET Flooding attacks through modified NetFPGA reference router. Asia NetFPGA Developers Workshop.

Kim, Y., & Kim, I. (2014). Involvers' behavior-based modeling in cyber targeted attack. SECUREWARE.

Ko, N., Noh, S., Park, J., Lee, S., & Park, H. (2010). An efficient anti-DDoS mechanism using flow-based forwarding technology. 9, 1-3.

Liao, Q., Li, H., Kang, S., & Liu, C. (2015). Application layer DDoS attack detection using cluster with label based on sparse vector

decomposition and rhythm matching. Security and Communication Networks, 8, 3111-3120.

Mirvaziri, H. (2017). A new method to reduce the effects of HTTP GET Flood attack. Future Computing and Informatics Journal, 2, 87-93.

Miu, T. W., Luo, D., & Wang, J. (2016). Modeling user browsing activity for application layer DDoS attack detection. 747-750.

Najafabadi, M., Khoshgoftaar, T., Calvert, C., & Kemp, C. (2017). User behavior anomaly detection for application layer DDoS attacks. Information Reuse and Integration (IRI), 154-161.

Praseed, A., & Thilagam, P. (2018). DDoS attacks at the application layer: Challenges and research perspectives for safeguarding Web applications (No. 2; IEEE Communications Surveys & Tutorials, pp. 661-685).

Ranjan, S., Swaminathan, R., Uysal, M., Nucci, A., & Knightly, E. (2008). DDoS-shield: DDoS-resilient scheduling to counter application layer attacks. IEEE/ACM Transactions on Networking, 17, 26-39.

Saravanan, R., Shanmuganathan, S., & Palanichamy, Y. (2016). Behavior-based detection of application layer distributed denial of service attacks during flash events. Turkish Journal of Electrical Engineering & Computer Sciences, 24, 510-523.

Shen, C., Cai, Z., Guan, X., Du, Y., & Maxion, R. (2013). User authentication through mouse dynamics. IEEE Transactions on Information Forensics and Security, 8, 16-30.

Singh, G., & Gupta, M. (2016). Distributed Denial-of-Service. Nternational Journal of Innovative Research in Science and Engineering, 2, 301-309.

Singh, K., Singh, P., & Kumar, K. (2017). Application layer HTTP-GET flood DDoS attacks: Research landscape and challenges. Computers & Security, 65, 344-372.

Singh, K., Singh, P., & Kumar, K. (2018). User behavior analytics based classification of application layer HTTP-GET flood attacks. Journal of Network and Computer Applications, 112, 97-114.

Spagna, S., Liebsch, M., Baldessari, R., Niccolini, S., Schmid, S., Garroppo, R., Ozawa, K., & Awano, J. (2013). Design principles of an operator owned highly distributed content delivery network. IEEE Communications Magazine, 51, 132-140.

Stevanovic, D., & Vlajic, N. (2014). Application-layer DDoS in dynamic Web-domains: Building defenses against next-generation attack behavior. Communications and Network Security (CNS), 490-491.

Thapngam, T., Yu, S., Zhou, W., & Beliakov, G. (2011). Discriminating DDoS attack traffic from flash crowd

through packet arrival patterns. 952-957.

Wang, J., Yang, X., Zhang, M., Long, K., & Xu, J. (2014). HTTP-SoLDiER: An HTTP-flooding attack detection scheme with the large deviation principle. Science China Information Sciences, 57, 1-15.

Xie, Y., & Yu, S. (2009). Monitoring the Application-Layer DDoS Attacks for Popular Websites. IEEE/ACM Transactions on Networking, 19, 5-25.

Yatagai, T., Isohara, T., & Sasase, I. (2007). Detection of HTTP-GET flood attack based on analysis of page access behavior. 232-235.

Yusof, A., Udzir, N., & Selamat, A. (2019). Systematic literature review and taxonomy for DDoS attack detection and prediction. International Journal of Digital Enterprise Technology, 3, 292-315.

Zhou, W., Jia, W., Wen, S., Xiang, Y., & Zhou, W. (2014). Detection and defense of application-layer DDoS attacks in backbone web traffic. Future Generation Computer Systems, 38, 36-46.

Zolotukhin, M., Kokkonen, T., Hämäläinen, T., & Siltanen, J. (2016). On application layer DDoS attack detection in high-speed encrypted networks. International Journal of Digital Content Technology and Its Applications (JDCTA), 10, 14-33.