Hindawi
Journal of Computer Networks and Communications
Volume 2019, Article ID 1283472, 10 pages
https://doi.org/10.1155/2019/1283472

Review Article

Review of Recent Detection Methods for HTTP DDoS Attack

Ghafar A. Jaafar, Shahidan M. Abdullah, and Saifuladli Ismail

Razak Faculty of Technology and Informatics, Universiti Teknologi Malaysia (UTM), 54100 Kuala Lumpur, Malaysia

Correspondence should be addressed to Ghafar A. Jaafar; afastars@gmail.com

Received 7 June 2018; Revised 9 November 2018; Accepted 12 December 2018; Published 10 January 2019

Academic Editor: Gianluigi Ferrari

With increment in dependency on web technology, a commensurate increase has been noted in destructive attempts to disrupt the essential web technologies, hence leading to service failures. Web servers that run on Hypertext Transfer Protocol (HTTP) are exposed to denial-of-service (DoS) attacks. A sophisticated version of this attack known as distributed denial of service (DDOS) is among the most dangerous Internet attacks, with the ability to overwhelm a web server, thereby slowing it down and potentially taking it down completely. This paper reviewed 12 recent detection of DDoS attack at the application layer published between January 2014 and December 2018. A summary of each detection method is summarised in table view, along with in-depth critical analysis, for future studies to conduct research pertaining to detection of HTTP DDoS attack.

1. Introduction

A second quarter security report produced by Kaspersky [1] indicated that the source attack of DDoS originated from 86 nations with attack duration up to 122 hours. The report illustrated increment in HTTP DDoS attack from 8.43% to 9.38%. Johnson Singh et al. [2] claimed that 540 Gbps DDoS attack occurred on 31st August 2016 against a federal government official website of Rio Olympic 2016 and the Ministry of Brazilian Sport. Based on the report produced by Arbor Networks (Worldwide Infrastructure Security Report (No. XII), 2017) [3] published in Q1 for the year 2017, attacks that occurred at the application layer were the most targeted, wherein 80% of the target attacked HTTP and 81% targeted at the Domain Name System (DNS).

Meng et al. [4] explained that execution of DDoS attacks at the application layer is complex to detect, because such attacks may be able to mimic a legal request with the purpose of using the system resources. A web server uses the HTTP and Hypertext Transfer Protocol Secure (HTTPS) protocols to process request from users. These protocols are widely used in commercial to operate business routines among banks, credit card payment gateways, government web servers, online shopping servers, social media servers, and broadcasting servers, to name a few. The consequence of the DDoS attack against a web server leads to monetary loss and loss of trust amongst people [5]. Najafabadi et al. [6] explained that the HTTP protocol is designed to have request and response, so as to allow communication to take place between client and web server.

This paper presents the recent detection methods of DDoS attack at the application layer and highlights several recommendations for future research. To the best of the authors' knowledge, no recent review has been produced regarding this topic. The rest of the paper is organised as follows: Section 2 describes DDoS attack at the application layer. Section 3 explains the types of DDoS attacks at application layers. Section 4 lists attack strategies performed by the attack. Section 5 elaborates web server architecture. Section 6 presents four defence techniques against DDoS. Section 7 depicts the recent detection methods of DDoS attack. Section 8 provides a critical analysis of the current detection, and lastly, the paper is concluded in Section 9.

2. DDoS Attack at Application Layers

DDoS attacks launched at the application layer pose challenges to detect as the request packet appears similar to the normal request packet [5, 7, 8]. Configuration and function related to application may lead to DDoS attack at the application layer [9], and the consequences of this attack may exhaust resources, such as network bandwidth, CPU

processing, and memory [10]. Jin et al. [11] explained that HTTP DDoS attacks occur when legitimate HTTP requests are initiated in large numbers.

DDOS attacks launched at the application layer require lower bandwidth to prevent legitimate users from surfing a web server, apart from mimicking traffic close to the authentic traffic [12]. The three factors that make DDOS detection difficult at the application layer are as follows [13]: (1) obscurity, HTTP protocol uses Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) connections to run its operation, hence the intricacy to differentiate legitimate from illegitimate traffic; (2) efficiency, HTTP DDOS attack only requires fewer connections to initiate a DDOS attack; and (3) lethality, the capability of the attack to overwhelm a web server immediately, thus resulting in service breakdown regardless of the type of hardware and its performance.

Protocol weaknesses at the application layer allow cyber intruders to exploit and execute malicious activities via HTTP, File Transfer Protocol (FTP), and telnet, among others [13]. DDoS attack at the application layer focuses on sending large amounts of GET request to a web server, and detection of this attack becomes more complicated when flash crowd is implemented. A flash crowd refers to the increasing number of legitimate HTTP GET request received by a web server due to several events, such as result announcements, new product launches, and breaking news [14]. Iyengar and Ganapathy [15] mentioned that flash crowds occur when plenty of authentic concurrent incoming connections are received by a web server in a short period of time. Ni et al. [10] explained that HTTP DDoS attacks occur due to higher request rate by a small group, while flash crowd increases the number of clients.

The source of DDoS is distributed as it is derived from various locations, including botnet participation to generate plenty of traffic against a server. Zargar et al. [12] explained the use of botnet in an HTTP communication, which poses challenge to track botnet structure called command and control. Botnet is created by using the HTTP protocol that dismisses command-and-control server, such as IRC-based botnet, since a web-based bot receives instruction periodically during a web request. Web-based botnets are stealthier as they can hide within authentic traffic. Botnets launched at the application layer are of two types: botnets that control and are configured by complicated Personal Home Page (PHP) script via protocol HTTP or HTTPS and web-based botnets that operate to report a website statistics [16]. Kolias et al. [17] explained in the light of Internet of Things (IoT) that popularity of IoT makes the devices a great tool to launch cyberattack. The IoT device is constantly connected to the Internet with naive security level, and these devices are vulnerable to botnet, thus may generate a vast number of DDOS attack traffics. The largest DDoS attack reported generated 1.2 TBps after using IoT devices, such as printer and camera home router, to launch the attack [18]. The existence of DDoS as services, such as boosters or stresses, simplifies the execution of the attack [19]. Besides, HTTP DDoS tools available for download without cost also contribute to cyberattack.

3. Types of DDOS Attack at Application Layer

Many studies have analysed (see [12, 13, 20–22]) the DDOS attack at the application layer based on the following categories.

3.1. Session Flooding Attack. Resources of a server become exhausted when session request rates get higher than valid users. This malicious activity may result in DDOS flooding attack, for instance, HTTP GET/POST flooding attack. In executing this attack, an attacker requires a large authentic HTTP request, and typically, a botnet is used as its ability to generate a valid request, commonly exceeding 10 requests in a second. This attack only requires a botnet to successfully initiate an attack.

3.2. Request Flooding Attack. This attack occurs when an attacker initiates a vast number of requests in one session. This request is larger than the request of a valid user. The HTTP GET/POST session is an instance of attack in this category that takes advantage from HTTP 1.1 feature, which allows more than one request within a single HTTP session. The structure of HTTP 1.1 allows the attacker to limit the HTTP session rate. The use of HTTP 1.1 also causes the attacker to bypass defence mechanism of session rate of a number of security systems. Rai and Challa [23] claimed that the botnet is used for this attack. The botnet is designed to have a command-and-control structure that allows cyber intruders to issue a command to botnet machines. The impacted machines are known as botnet-listed in command-and-control server as they give instruction to the botnet to launch HTTP GET flood. This attack can exhaust server resources as the botnet sends plenty of HTTP GET flood requests to a server.

3.3. Asymmetric Attack. Cyber intruders use HTTP session that contains high workload of requests, which is generated by downloading huge files or excessive running queries from a database server.

3.4. Slow Request/Response Attack. An attacker sends high workload of requests to initiate attack in the form of a session. The consequence of this attack introduces inaccessibility against a server as the attacker partially sends HTTP requests that grow quickly and repeatedly, update slowly, and never close. This continuous attack will make an available socket of a server to be full due to these requests. Another example of this attack is HTTP fragmentation, where the connection of HTTP is held for some time to bring down the server. Rai and Challa [23] asserted that the attack operates under a threshold limit to complicate the victim with malicious traffic that resembles legitimate traffic. The Slowloris attack is the example from this attack category, and it works by sending a large amount of simultaneous HTTP requests, be it GET or POST, to a server. A server will continuously open separate connections as each HTTP request fails to complete its connection. The consequence of

this attack denies users from gaining connection to a server as the server concurrent connection is exhausted.

4. DDOS Attack Strategies at Application Layer

Singh et al., [9] categorised HTTP DDOS attacks into several subclasses, as follows.

4.1. Server Load. The attacker uses botnet to continuously send malicious requests against a web server aggressively, hence causing the server to drop legitimate requests as the web server resources run out.

4.2. Increasing. The attacker uses the low request value to initiate attack and slowly raises the value. This behaviour of attack is difficult to detect as malicious HTTP traffic does not send requests aggressively to the server during the attack.

4.3. Constant. Cyber intruders must specify a number of request rates to be sent to the victim HTTP web server. The request number is called constant as it has the same number as when the botnet sends malicious requests to a web server.

4.4. Target Web Page. HTTP DDOS attack occurs at single and multiple web pages, where the attacker imitates legitimate users access pattern to deceive attack detection. The web pages are accessed by botnets that mimic the human access pattern.

4.5. Single Web Pages Attack. The attacker uses a single web page that belongs to a website. The botnet continuously send malicious HTTP requests to the web server.

4.6. Main Page Attack. Cyber intruders specifically focus on the main page of the websites to deny access among legitimate users. The traffic botnet is used to repeatedly send malicious requests to the HTTP server. The impact of this attack only occurs on the main page of the website, while the subpages of the website are not affected.

4.7. Dominant Page Attack. Cyber intruders figure out web pages that are sought by legitimate users for access. The attacker then focuses on that page to execute HTTP DDOS attacks in order to prevent a legitimate user from accessing the greater interest of the web pages. This attack only affects web pages with greater interest for users to browse.

4.8. Multiple Page Attack. A cyber attacker initiates the attack at multiple web pages from a website. This technique avoids detection as the malicious HTTP request imitates a human access pattern. For instance, a human will open more than one web page to find information while surfing a website. During this attack, pages that exist in the website will not be accessible as the attacker targets multiple web pages.

4.9. Reply Flood Attack. The botnet command by the attacker sends an HTTP traffic at an inflated rate to gain a resource of the web server to prevent the web server to surf legitimate HTTP request. The attacks work by gaining human access pattern to prohibit the detection system from blocking the malicious request.

4.10. Rare Change Page Attack. The common structure of a web system will group a page into a specific group to make the page content more structured and user-friendly. Since the arrangement of the web page is grouped, the attacker may compromise the group page by commanding botnet to that web page. Because the web page is designed in the group, the page becomes the most targeted group. This attack prevents a user from opening a web page that belongs to a specific group.

4.11. Frequent Changes Attack. An attacker performs attack into a web page that belongs to different categories. This attack will rotate the sent malicious request to distinct web page categories. This attack only affects specific categories of the website, as other web pages are still accessible during the attack.

4.12. Hot Pages Attack. Each web-based system will have frequent open pages. Hence, cyber intruders attack the most visited pages to prevent legitimate users from accessing, as the main objective for the DDOS attack is to avoid users from opening the pages.

4.13. Web Proxy Attack. The attacker uses the proxy server as a mediator to generate attack traffic. The use of the proxy server to generate attack traffic introduces difficulties in detecting the source attack. Multiple proxy servers are used to generate plenty of HTTP requests to overwhelm the web server.

5. Web Server Architecture

Clients' requests for online services initiate an HTTP GET request to a server. Prior to this, a TCP connection must be established before a client can successfully obtain response from a web server. Singh et al. [9] listed the processes involved in HTTP GET requests. First, a web server listens to an incoming connection, including TCP connection, as this connection must be established prior to other stages. In the second stage, socket queue, which is responsible for holding the entire HTTP GET request until a dedicated thread, is assigned to serve the request. Third, the request queue is accountable to process and respond to individual request. Upon completing these processes, the web server sends an HTTP response. During the HTTP GET flood attacks, the request queue becomes full immediately, thus dropping the incoming requests sent by authentic users. Figure 1 illustrates the typical architecture of a web server during HTTP GET request and response.

A web server does not perform filtering to determine if an HTTP GET request is genuine or fake [9]. During the

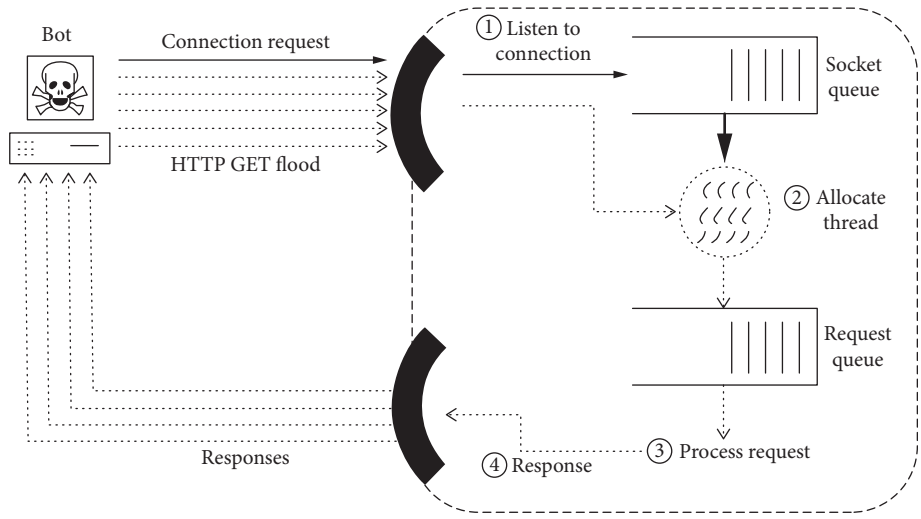

Figure 1: Web server architecture.

HTTP GET flood attack, a web server will continuously receive and process the request as it assumes the request is from authentic sources. The continuity of such requests at a higher rate will collapse a web server as it is unable to process a valid HTTP GET request. Beitollahi and Deconinck [24] and Sree and Bhanu [25] mentioned that an HTTP GET flood attack can stress bandwidth and outbound traffic, memory, CPU cycles, and input and output devices.

6. HTTP DDoS Detection

There are several phases that are involved in defencing DDoS attack, and prior studies [20, 26] explained four phases involved as follows.

6.1. Prevention. The prevention phase focuses on protecting a system against an attack by applying appropriate security appliances at varied places. Besides that, prevention also protects server resources and ensures that online services are ready to surf the genuine client. DDoS attacks launched through automated tools allows several programs to access certain web pages without human intervention. Possible prevention against this type of attack through website design is to allow only the authentic user to access web server services and resource. Web design should be efficient that could not be delayed by the attacker.

6.2. Mitigation. The mitigation phase is applied when an attack occurs, and a suitable security countermeasure is executed to handle the attack or to slow down the attack. A mitigation technique operates by stopping the attack. Formation of DDoS mitigation is considered better when the attack traffic recognised as normal is minimal, which is also known as false positive rate. Apart from that mitigation technique supposed to block a source IP address of illegitimate traffic that generates the attack, this process will directly guarantee the authentic client to be able to access a web service.

6.3. Detection. The detection phase requires analysis of the running system to discover malicious traffic that leads to DDoS attack. Detection involves a sophisticated approach to identify large illegal GET request traffic against a web server. Most of the detection techniques were applied to form DDoS detection known as pattern matching, clustering, statistical methods, deviation analysis, associations, and correlation. Formation of detection usually employs data history as the main source to train the data to generate a threshold which will be assigned to a parameter via a specific method to count the GET request received. The false positive rate refers to incorrect classification of attack traffic predicted as genuine, and effective DDoS detection generates a minimal false positive rate.

6.4. Monitoring. As for the monitoring phases, necessary information about a host or network is obtained by using tools, such as network monitoring software. Monitoring is conducted in real time as it becomes compulsory for detection of DDoS attack. A process of monitoring becomes complicated when the attacker utilised botnet that is situated at multiple locations around the world to launch DDoS attack at a minimal rate. According to [27], dynamic monitoring is required in order to constitute defences for attacks. Figure 2 presents the graphical view of the defence life cycle.

7. Recent Detection Methods for HTTP DDoS Attack

This section focuses on the recent HTTP DDoS detection techniques proposed and applied since the past five years based on the published work.

Hameed and Ali [19] introduced a framework called HADEC to detect live high-rate DDoS attack that occurs at network and application layers, such as TCP-SYN, HTTP GET, UDP, and ICMP. The framework is composed of two main components: detection server and capture server. Live DDoS detection begins by capturing the server that is responsible to capture live network traffic and transfer the log

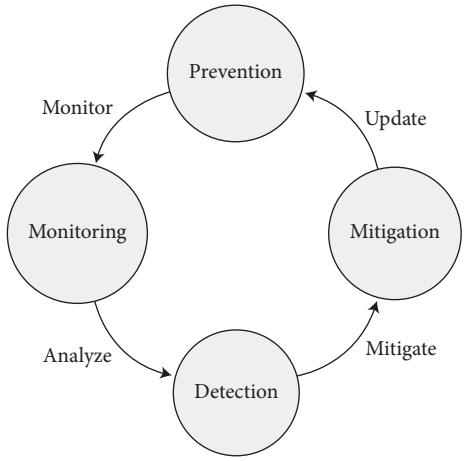

Figure 2: DDoS defence life cycle.

to the detection server for processing. The detection calculates incoming packet for UDP, ICMP, and HTTP to detect an attack if the source connection exceeds the predefined threshold. The proposed detection provides low-cost solutions for financial institution, as well as small and medium companies.

Behal et al. [28] proposed a detection method called D-FACE to detect four traffic types: legitimate user, low-rate, high-rate, and flash event traffic. The detection employs entropy difference that contains normal traffic flow, while the value of source IP entropy is the detection matrix to reckon the attack. The detection begins with extraction of the related header that classifies the network into a unique network flow. Segregation of low-rate, high-rate, and flash event traffic is based by comparing the current incoming traffic rate in each time window and based on information traffic value.

Singh et al. [29] introduced a method that detects HTTP DDoS attack via the machine-learning approach to distinguish botnet from legitimate users in detecting attack traffic, authentic traffic, and flash traffic. The proposed system is deployed as proxy and performs inspection against user behaviour instead of monitoring the entire traffic. The proposed work detects the botnet source and examines user behaviour to detect malicious request against the web server.

Sreeram and Vuppala [30] proposed a machine-learning matrix with a bio-inspired bat algorithm to allow fast and early detection of HTTP DDoS attacks. The work incorporated time intervals, instead of user sessions, and packet patterns to generate a detection algorithm. The time interval uses machine-learning matrix by assigning a value of maximum sessions for one-time interval and computing a number of sessions in one-time interval to detect DDoS attack at application layers. The matrix also accounts for two pages of HTTP GET request. The frequency of a web page accessed by users and the time gap between first page request and second page are determined to monitor user behaviour.

Aborujilah and Musa [31] introduced cloud-based detection of HTTP DDoS by using statistical approach with the covariance matrix. The detection introduced two algorithms known as training and testing to recognise a different type of HTTP flooding attack based on attack behaviour. A training algorithm was used to construct normal patterns of network traffic, and the testing algorithm was used to determine the types of traffic received. The outcomes obtained from this research had been evaluated by using the confusion matrix to measure detection performance and provide results of internal and external cloud environment.

Singh and De [32] employed multilayer perceptron with a genetic algorithm (MLP-GA) to detect HTTP DDoS attack. The proposed detection utilised four parameters to generate detection at application layers. A normal user has specific time interval, as an authentic user searches and reads when accessing a web page and when moving to other pages. The detection technique suggested by the researchers counts the number of HTTP GET requests received by the web server and calculates the number of IP addresses targeting the server over 20 seconds. The proposed detection also inspects the port number used by HTTP DDoS, as ports used by HTTP DDoS attackers are varied and remain open. The detection method employed fixed frame length to conduct detection, as according to these researchers, HTTP DDoS attackers employ static protocol lengths.

Hoque et al. [33] proposed a method for detection of DDoS at the application layer in real time at the victim end. The proposed work utilised software and hardware adopted from the framework created to distinguish normal from fake traffic in real time. The three main components incorporated into the framework were preprocessor, hardware module, and security manager, which processed source IPs, source IPs index variation, and packet rate, to detect the attack.

Johnson Singh et al. [2] introduced a detection scheme to reckon high and low rates of DDoS attack. The detection was performed by computing a number of HTTP GET request, entropy, and variance for each connection. The HTTP GET requests were counted within 20 s time window.

Liao et al. [34] proposed a detection technique based on user access frequencies, specifically focusing on request time interval and frequency request to detect DDoS attacks at the application layer. The time interval refers to the present and the next HTTP GET requests. The time interval for a standard user may be larger when compared to those of an attacker, as a normal user will spend more time browsing interesting pages. For instance, the time interval for normal browsing is 246 seconds for the first page and around 572 seconds to open the next page. However, in the case of DDoS attacks, the time interval for the present and subsequent requests is shorter.

Shiaeles and Papadaki [35] introduced multilayer IP spoofing to detect application layer DDoS attack against the web server. The proposed detection method is called fuzzy hybrid spoofing detection (FHSD) and used source MAC address, hop count, geographical of IP address, operating system (OS) passive fingerprinting, and web browser user agents. In order to decrease false positive and false negative, cross inspection was performed via operating passive fingerprinting and HTTP user agents. Passive operating fingerprinting and HTTP user agents were used to confirm the name of the OS used by a client. The proposed work also

detects botnet in a local network using MAC address and IP paring. The limitation of the proposed work refers to the database that stores information required to update daily for better results.

Wang et al. [36] proposed a detection technique for HTTP flooding attacks based on web browsing clicks called HTTP soldier. It displayed the ability to distinguish normal user from malicious one through the use of large-deviation probability. The technique highly relied on web page popularity to detect HTTP flood attack. The operation of HTTP soldier used a predefined threshold to be compared with large-deviation probability. The large-deviation probability may affect some normal users. The researchers clearly mentioned that a single URL attack was ineffective to detect by using their technique. This study employed simulation software to evaluate their proposed detection and only measured false-positive rate.

Nam and Djuraev [37] proposed a detection technique based on the workload of the source node. This technique used multiple levels to protect a web server. The first layer allowed or rejected a received connection by inspecting the source IP address with the whitelist. The registered IP addresses were allowed to establish a connection with web servers to obtain service, while the connection of nonregistered IP addresses was dropped. The allowed IP addresses were inspected, and if they behaved maliciously, the connection would be dropped, and the IP addresses would be blacklisted. The researchers stressed that their work has limitation when legitimate user access a server for streaming services or when downloading huge file, as this can lead to false positive. The result of this study sheds light on server response time. The outcome shows that the proposed defence system detects DDoS attacks after 90 seconds and the server response time goes back to normal upon detection of attack.

8. Critical Analysis

8.1. Results. Based on the results reported by each author, only Sreeram and Vuppala [30] evaluated their proposed solution by using a complete matrix of detection (e.g., true positive, true negative, false positive, false negative, precision, recall, specificity, and accuracy). This review found that Aborujilah and Musa and Singh and De [31, 32] only employed true and false detection matrix to assess their proposed methods. Meanwhile, other studies were uncertain as some evaluated detection rate, false positive, and accuracy, failed in providing a full detection matrix, as explained above. This constraint must be prevented to ensure that the proposed work is ready to be deployed in producing environment, apart from having the ability to work precisely in detecting all attack scenarios.

A complete detection matrix allows future researches to improve detection performance, hence resulting in full detection matrix that should be executed and shared publicly. Implementation of the confusion matrix that measures detection performance is the best option as this matrix has a wide range of measurement of detection. The matrix has also been utilised by many researchers [38–45].

8.2. Dataset. The experiment datasets utilised by prior scholars were mixed, while some relied only on one dataset [30, 36, 37]. The use of only one dataset as reference (e.g., benchmark and analysis) seems insufficient as HTTP DDoS attack patterns are varied. Further analysis performed [2, 28, 29, 31–35] took into account more than one dataset. Nevertheless, the used datasets were obsolete to perform comparison and analysis. Critical inspection against researchers who employed more than one dataset showed that they also executed their experiments to obtain the dataset. Old datasets should be avoided as they contain obsolete and meaningless data [9], while retrieving real cutting-edge attack dataset is challenging as they are unavailable publicly [46].

8.3. Self-Generate Dataset. Future direction for detection of HTTP DDoS attack should focus on the self-generate dataset by utilising real HTTP tools. The actual tools are available for download and can be used for multiple purposes such as investigation and evaluate a proposed work. The self-generate dataset is a solution that can be deployed by future researches in resolving issues related to old dataset and not publicly share. There are many DDoS tools, such as HOIC, LOIC, HULK, Rudy, DDOSIM, Bonesi, PyLoris, XOIC, and Slowloris, highlighted and utilised by prior researchers [2, 20, 32, 35].

8.4. Evaluation Method. This review also found that a few researchers evaluated their proposed work by using simulation software [30–32, 36]. On the other hand, some [2, 19, 28, 33–35] performed real experiments to evaluate their outcomes and found only one study [37] assessed their work by using simulation software and experimental works.

A proposed work should be evaluated by considering a wide range of network architectures and potential attack strategies that may be utilised by an attacker for detection purpose. The good evidence for this statement can be supported by studies performed by Singh et al. [9], who mentioned that network address translation (NAT) and web proxies led to complexity and could result in incorrect detection outcomes. The evaluation of a proposed work should weigh in a wide range of network designs, along with lists of potential attack strategies, so as to ensure that the proposed detection method has the ability to work outside the academic world and not merely for education purposes. Singh et al. [9] outlined a list of approaches used by attackers to launch DDoS against web pages of online services, which may serve as reference for future research. In order to achieve this goal, several challenging factors must be faced, such as time constraint, knowledge about network and security, hardware and software to purchase, and viable configuration.

8.5. Detection Method for Future Work. Future research studies should focus on providing solutions that are practical for implementation in the actual environment, apart from using real HTTP DDoS tools when evaluating their proposed detection techniques, so as to offer benefits to both the parties, academic and industry. This is because a proposed

Table 1: Summary of recent detection techniques of HTTP DDoS attack.

No.	Name	Parameter	DDoS detection level	Evaluation method	Dataset	Performance matrix
1	HADEC: hadoop-based live DDoS detection framework [19]	Timestamps, source IP, destination IP, packet protocol, and packet header	High-rate DDoS: TCP-SYN, HTTP GET, UDP, and ICMP	Experiment	Experiment dataset	Measure utilisation, CPU, and Memory
2	D-FACE: an anomaly-based distributed approach for early detection of DDoS attacks and flash events [28]	Time window size, packet header, and generalised parameter	High-rate and low-rate DDoS attack and flash crowd	Experiment	MIT Lincoln, CAIDA, and FIFA	accuracy, false-positive rate classification rate, F-measure, and precision
3	User behaviour analytics-based classification of application layer HTTP GET flood attacks [29]	Request index, response index, popularity index, repetition index, and classifier algorithms	High-rate DDoS attack	Experiment	WorldCup98, Clarknet, and NASA	True positive, true negative, false positive, and false negative
4	HTTP flood attack detection in the application layer using machine-learning metrics and bio-inspired bat algorithm [30]	Time frame length, maximum number of sessions (ms), page access count (pac), minimum time interval between two pages (mti), and packets observed per each type of packet (PC)	High-rate DDoS attack	Simulation software	CAIDA	True positive, false positive, true negative, false negative, precision, recall, specificity, accuracy, and F-measure
5	Cloud-based DDoS HTTP attack detection using covariance matrix approach [31]	TCP packet header and Covariance matrix	High-rate DDoS attack	Simulation (MATLAB)	KDD cup 99 and experiment dataset	Detection rate, false positive, false negative, accuracy, error rate, and AUC
6	MLP-GA-based algorithm to detect application layer DDoS attack (Singh and De [32])	Number of HTTP count, number of IP addresses, constant mapping function, and fixed frame length	Low-rate DDoS attack	Simulation software	EPA-HTTP, CAIDA 2007, and experiment dataset	Accuracy, false positive, false negative, true positive, and true negative
7	Real-time DDoS attack detection using FPGA [33]	Source IPs, Source IPs index variation, and packet rate	High-rate HTTP DDoS	Experiments	CAIDA, TUIDS, and DARPA	Accuracy, detection rate, false positive, and false negative
8	Entropy-based application layer DDoS attack detection using artificial neural networks [2]	HTTP GET request count per connection, IP address variance, HTTP GET request counts, and multilayer perceptron with genetic machine-learning algorithm (MLP-GA)	High-rate DDoS attack	Experiments	Standard EPA-HTTP, experiment dataset, CAIDA 2007, DARPA 2009, and BONESI-generated datasets	Accuracy, sensitivity, and specificity
9	Application layer DDoS attack detection using cluster with label based on sparse vector decomposition and rhythm matching [34]	Request interval sequence part, and request frequency sequence part	High-rate DDoS attack	Experiments	ClarkNet HTTP, and experiment dataset	Accuracy, detection rate, and false positive
10	FHSD: an improved IP spoof detection method for web DDoS attacks [35]	Source MAC address, hop count, GeoIP, OS passive fingerprinting, and web browser user agent	High-rate DDoS attack	Experiments	DARPA LLDOS inside 1.0 and experiments dataset	Detection rate
11	HTTP soldier: an HTTP flooding attack detection scheme with the large-deviation principle [36]	Threshold exponentially weigh moving average algorithm Large deviation probability theory	High-rate DDoS attack	Simulation (NS3)	University web logs	False positive
12	Defending HTTP web servers against DDoS attacks through busy period-based attack flow detection [37]	Threshold whitelist and blacklist	High-rate DDoS attack	Simulation (OPNET experiment)	Experiment dataset	Detection speed

detection method should not only work to achieve an academic target but also offer an option for the world cyber security in detecting HTTP DDoS attacks. The proposed solutions to DDoS are academic interest, and only some have been implemented in real time [47, 48]. The use of real tools of HTTP DDoS attack will help to gain input about the current attack strategies and prediction about a future attack.

8.6. Detection Techniques. The parameter used to generate detection is verities. This review revealed that the use of the right parameter is important to ensure that the proposed detection method is able to detect HTTP DDoS attack. This review also discovered that the detection method of HTTP DDoS is built based on three main elements: (1) technique, e.g., IP spoofing; (2) parameter, e.g., TCP header; and (3) flow, e.g., operation flow [2, 30–32, 34–37]. A component selection for the three elements mentioned above is crucial as it leads to detection quality, e.g., true positive, true negative, false positive, false negative, precision, recall, specificity, and accuracy. A future researcher has to consider the main elements stated above to ease formation of the proposed detection technique.

8.7. Level of Attack. Based on the review presented in Table 1, no study has proposed a solution that is able to detect three types of DDoS attacks: flash crowd, high-rate, and low-rate DDoS attack. Most studies focused on detecting HTTP DDoS at the high rate, while only one researcher focused on low-rate HTTP DDoS attack. The detection scope should extend to cater to all types of HTTP DDoS, as the techniques reviewed in this paper tend to propose solutions in a segregated manner. A list of researchers specified DDoS attack at application down to several categories: request flooding, session flooding, and asymmetric and slow request [12, 13, 20–22]. This appears to be a great indicator for future research studies to look deep down the attack patterns.

8.8. Learn Programming and Explore Attack Codes. HTTP DDoS tools, such as GoldenEye, UFONET, Wreckquest, and HULK, are available for download, and they are written in the python programming language. Thus, learning programming language can bring us to a step forward in detecting HTTP DDoS and prediction of future attacks by exploring the attack code with consideration of DDoS attack strategies, as outlined by Singh et al. [9]. Exploration of the code enhances our understanding towards the behaviour of HTTP DDoS, apart from discovering appropriate strategies to counter the attack.

8.9. Academic and Industry. Academic and industry have to cooperate in sharing attack logs for academic purposes. Behal and Kumar [49] conducted a study related to DDoS using prior dataset and drew some limitations against publicly available dataset as the dataset was captured from the network layer and had concealed information about the application layer. Hence, an industry that receives real dataset or received HTTP DDoS attack should filter and remove private data related to its web server so as to ensure no data leakage. The attack pattern would give direct impact as a researcher is able to write HTTP DDoS code based on the pattern and to seek solution, thus highlighting the importance of programming.

8.10. Mixed HTTP Traffic. HTTP DDoS attack has a number of attack strategies to mix the traffic to be more complex for detection; thus, the future work to detect DDoS attack should cater varied types of attack strategies, such as attack that comes from proxy, botnet, and web crawler simultaneously. Besides, the use of the IoT device may lead to worse circumstance of HTTP DDoS as the attacker may utilise a botnet that originates from such devices to recruit a cyber army to launch attack against a web server. Table 1 indicates that all the proposed solutions catered to only the HTTP protocol that applied port 80 to detect an attack. Hence, future research studies should consider HTTPS protocol detection in an in-depth manner. Zolotukhin et al. [50] explained that most of the latest studies seemed to focus on HTTP DDoS attack, while leaving detection of HTTPS protocol for DDoS in the dark.

9. Conclusion

This paper presents a review of the recent detection methods in recognising DDoS attack at the application layer. Researches related to DDoS attack have gained much attention, particularly those that occur at the application layer. The detection of DDoS attack is rather challenging as the traffic has the ability to mimic the genuine GET request. Due to many forms of devices that can be affected by botnet, such as IoT devices and existence of DDoS as services, the detection of such attack can become significantly complex. The recent techniques employed to detect HTTP DDoS attack have produced various approaches for detection purposes. However, it should be noted that challenges need to be identified and overcome for different types of DDoS attack strategies. The critical analysis has outlined several points that one would need to give closer attention for future research.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This review paper was compiled at University Technology Malaysia (UTM).

References

[1] Kaspersky, *DDoS attacks in Q2 2017*, 2017, https://securelist.com/ddos-attacks-in-q2-2017/79241/.

[2] K. Johnson Singh, K. Thongam, and T. De, "Entropy-based application layer DDoS attack detection using artificial neural networks," *Entropy*, vol. 18, no. 10, p. 350, 2016.

[3] Worldwide Infrastructure Security Report (No. XII), 2017, https://pages.arbornetworks.com/rs/082-KNA-087/images/12th_Worldwide_Infrastructure_Security_Report.pdf.

[4] B. Meng, W. Andi, X. Jian, and Z. Fucai, "DDOS attack detection system based on analysis of users' behaviors for application layer," in *Proceedings of IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, Guangzhou, China, July 2017.

[5] H. Beitollahi and G. Deconinck, "Analyzing well-known countermeasures against distributed denial of service attacks," *Computer Communications*, vol. 35, no. 11, pp. 1312–1332, 2012.

[6] M. M. Najafabadi, T. M. Khoshgoftaar, C. Calvert, and C. Kemp, "User behavior anomaly detection for application layer ddos attacks," in *Proceedings of 2017 IEEE International Conference on Information Reuse and Integration (IRI)*, San Diego, CA, USA, August 2017.

[7] K. Subramanian, P. Gunasekaran, and M. Selvaraj, "Two layer defending mechanism against DDoS attacks," *International Arab Journal of Information Technology (IAJIT)*, vol. 12, no. 4, 2015.

[8] X. Yuan, C. Li, and X. Li, "DeepDefense: identifying DDoS attack via deep learning," in *Proceedings of 2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, Hong Kong, China, May 2017.

[9] K. Singh, P. Singh, and K. Kumar, "Application layer HTTP-GET flood DDoS attacks: research landscape and challenges," *Computers and Security*, vol. 65, pp. 344–372, 2017.

[10] T. Ni, X. Gu, H. Wang, and Y. Li, "Real-time detection of application-layer DDoS attack using time series analysis," *Journal of Control Science and Engineering*, vol. 2013, pp. 1–6, 2013.

[11] W. Jin, Z. Min, Y. Xiaolong, L. Keping, and X. Jie, "HTTP-sCAN: detecting HTTP-flooding attack by modeling multi-features of web browsing behavior from noisy web-logs," *China Communications*, vol. 12, no. 2, pp. 118–128, 2015.

[12] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013.

[13] N. Hoque, D. K. Bhattacharyya, and J. K. Kalita, "Botnet in DDoS attacks: trends and challenges," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 4, pp. 2242–2270, 2015.

[14] R. Saravanan, S. Shanmuganathan, and Y. Palanichamy, "Behavior-based detection of application layer distributed denial of service attacks during flash events," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 24, pp. 510–523, 2016.

[15] N. C. S. N. Iyengar and G. Ganapathy, "An effective layered load balance defensive mechanism against DDoS attacks in cloud computing environment," *International Journal of Security and Its Applications*, vol. 9, no. 7, pp. 17–36, 2015.

[16] E. Alomari, S. Manickam, B. Gupta, S. Karuppayah, and R. Alfaris, "Botnet-based distributed denial of service (DDoS) attacks on web servers: classification and art," 2012, http://arxiv.org/abs/1208.0403.

[17] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[18] N. Woolf, "DDoS attack that disrupted internet was largest of its kind in history, experts say, The Guardian," October 2016, https://www.theguardian.6com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet.

[19] S. Hameed and U. Ali, "HADEC: hadoop-based live DDoS detection framework," *EURASIP Journal on Information Security*, vol. 2018, no. 1, p. 11, 2018.

[20] V. Kumar and K. Kumar, "Classification of DDoS attack tools and its handling techniques and strategy at application layer," in *Proceedings of International Conference on Advances in Computing, Communication, and Automation (ICACCA)(Fall)*, pp. 1–6, Dehradun, India, September 2016.

[21] S. Ranjan, R. Swaminathan, M. Uysal, A. Nucci, and E. Knightly, "DDoS-shield: DDoS-resilient scheduling to counter application layer attacks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 1, pp. 26–39, 2009.

[22] S. Yadav and S. Subramanian, "Detection of application layer DDoS attack by feature learning using stacked AutoEncoder," in *Proceedings of 2016 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT)*, New Delhi, India, March 2016.

[23] A. Rai and R. K. Challa, "Survey on recent DDoS mitigation techniques and comparative analysis," in *Proceedings of 2016 Second International Conference on Computational Intelligence & Communication Technology (CICT)*, pp. 96–101, Ghaziabad, India, February 2016.

[24] H. Beitollahi and G. Deconinck, "ConnectionScore: a statistical technique to resist application-layer DDoS attacks," *Journal of Ambient Intelligence and Humanized Computing*, vol. 5, no. 3, pp. 425–442, 2013.

[25] T. R. Sree and S. M. S. Bhanu, "HADM: detection of HTTP GET flooding attacks by using Analytical hierarchical process and Dempster-Shafer theory with MapReduce," *Security and Communication Networks*, vol. 9, no. 17, pp. 4341–4357, 2016.

[26] A. Shameli-Sendi, M. Pourzandi, M. Fekih-Ahmed, and M. Cheriet, "Taxonomy of Distributed Denial of Service mitigation approaches for cloud computing," *Journal of Network and Computer Applications*, vol. 58, pp. 165–179, 2015.

[27] X. Yi and Y. Shun-Zheng, "Monitoring the application-layer DDoS attacks for popular websites," *IEEE/ACM Transactions on Networking*, vol. 17, no. 1, pp. 15–25, 2009.

[28] S. Behal, K. Kumar, and M. Sachdeva, "D-FACE: an anomaly based distributed approach for early detection of DDoS attacks and flash events," *Journal of Network and Computer Applications*, vol. 111, pp. 49–63, 2018.

[29] K. Singh, P. Singh, and K. Kumar, "User behavior analytics-based classification of application layer HTTP-GET flood attacks," *Journal of Network and Computer Applications*, vol. 112, pp. 97–114, 2018.

[30] I. Sreeram and V. P. K. Vuppala, "HTTP flood attack detection in application layer using machine learning metrics and bio inspired bat algorithm," *Applied Computing and Informatics*, 2017, in press.

[31] A. Aborujilah and S. Musa, "Cloud-based DDoS HTTP attack detection using covariance matrix approach," *Journal of Computer Networks and Communications*, vol. 2017, Article ID 7674594, 8 pages, 2017.

[32] K. J. Singh and T. De, "MLP-GA based algorithm to detect application layer DDoS attack," *Journal of Information Security and Applications*, vol. 36, pp. 145–153, 2017.

[33] N. Hoque, H. Kashyap, and D. K. Bhattacharyya, "Real-time DDoS attack detection using FPGA," *Computer Communications*, vol. 110, pp. 48–58, 2017.

[34] Q. Liao, H. Li, S. Kang, and C. Liu, "Application layer DDoS attack detection using cluster with label based on sparse vector decomposition and rhythm matching," *Security and Communication Networks*, vol. 8, no. 17, pp. 3111–3120, 2015.

[35] S. N. Shiaeles and M. Papadaki, "FHSD: an improved IP spoof detection method for web DDoS attacks," *Computer Journal*, vol. 58, no. 4, pp. 892–903, 2014.

[36] J. Wang, X. Yang, M. Zhang, K. Long, and J. Xu, "HTTP-SoLDiER: an HTTP-flooding attack detection scheme with the large deviation principle," *Science China Information Sciences*, vol. 57, no. 10, pp. 1–15, 2014.

[37] S. Y. Nam and S. Djuraev, "Defending HTTP web servers against DDoS attacks through busy period-based attack flow detection," *KSII Transactions on Internet and Information Systems*, vol. 8, no. 7, pp. 2512–2531, 2014.

[38] A. R. Kang and A. Mohaisen, "Automatic alerts annotation for improving DDoS mitigation systems," in *Proceedings of 2016 IEEE Conference on Communications and Network Security (CNS)*, Philadelphia, PA USA, October 2016.

[39] S. Lakshminarasimman, S. Ruswin, and K. Sundarakantham, "Detecting DDoS attacks using decision tree algorithm," in *Proceedings of 2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN)*, Chennai, India, March 2017.

[40] Q. Liao, H. Li, S. Kang, and C. Liu, "Feature extraction and construction of application layer DDoS attack based on user behavior," in *Proceedings of 2014 33rd Chinese Control Conference (CCC)*, Nanjing, China, July 2014.

[41] I. Mukhopadhayay, S. Polle, and P. Naskar, "Simulation of denial of service (DoS) attack using matlab and xilinx," *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 16, no. 3, pp. 119–125, 2014.

[42] J. D. Ndibwile, A. Govardhan, K. Okada, and Y. Kadobayashi, "Web server protection against application layer DDoS attacks using machine learning and traffic authentication," in *Proceedings of 2015 IEEE 39th Annual Computer Software and Applications Conference*, Taichung, Taiwan, July 2015.

[43] D. Peraković, M. Periša, I. Cvitić, and S. Husnjak, "Artificial neuron network implementation in detection and classification of DDoS traffic," in *Proceedings of 2016 24th Telecommunications Forum (TELFOR)*, Belgrade, Serbia, November 2016.

[44] K. J. Singh and T. De, "Analysis of application layer DDoS attack detection parameters using statistical classifiers," *Internetworking Indonesia*, vol. 9, no. 2, pp. 23–31, 2017.

[45] I. Sofi, A. Mahajan, and V. Mansotra, "Machine learning techniques used for the detection and analysis of modern types of DDoS attacks," *Learning*, vol. 4, no. 6, 2017.

[46] S. Yadav and S. Selvakumar, "Detection of application layer DDoS attack by modeling user behavior using logistic regression," in *Proceedings of 2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions)*, Noida, India, September 2015.

[47] S. Behal and K. Kumar, "Detection of DDoS attacks and flash events using novel information theory metrics," *Computer Networks*, vol. 116, pp. 96–110, 2017.

[48] S. Behal, K. Kumar, and M. Sachdeva, "Characterizing DDoS attacks and flash events: review, research gaps and future directions," *Computer Science Review*, vol. 25, pp. 101–114, 2017.

[49] S. Behal and K. Kumar, "Trends in Validation of DDoS Research," in *Proceedings of International Conference on Computational Modeling and Security (CMS 2016)*, Bengaluru, India, February 2016.

[50] M. Zolotukhin, T. Hämäläinen, T. Kokkonen, and J. Siltanen, "Increasing web service availability by detecting application-layer DDoS attacks in encrypted traffic," in *Proceedings of 2016 23rd International Conference on Telecommunications (ICT)*, pp. 1–6, Thessaloniki, Greece, February 2016.