A Review of Detection Challenge for Signature and Anomaly-Based Detection in Detecting HTTP DDoS Attacks

Article history

Received:
14 August 2025

Received in revised form:
1 September 2025

Accepted:
2 November 2025

Published online:
26 December 2025

*Corresponding author
abdulghafar@utm.my

Abdul Ghafar Jaafar[1], Nur Hanis Sabrina Suhaimi[2], Abdulrahman Aminu Ghali[3], Hafizah Mansor[4], Ganthan Narayana Samy[5], Nazri Kama[6], Noor Hafizah Hassan[7]

[1,5,6,7]*Faculty of Artificial Intelligence, Universiti Teknologi Malaysia (UTM), 54100 Kuala Lumpur, Malaysia*

[2]*Faculty of Information Science & Technology Universiti Kebangsaan Malaysia, Bangi, Malaysia*

[3]*Faculty of Information and Communication Technology (FICT), Universiti Tunku Abdul Rahman, 31900 Kampar, Malaysia*

[4]*Kulliyyah of ICT, International Islamic University Malaysia, Selangor, Malaysia*

Abstract

Distributed Denial of Service (DDoS) attacks have become one of the most serious concerns in the cybersecurity domain due to their ability to mimic legitimate traffic. The attack is significantly challenging to detect when occurring at the application layer because it exploits genuine request patterns, forged headers, automated attack tools, and public proxies to mimic legitimate traffic, making detection extremely difficult. This paper reviews signature-based and anomaly-based detection techniques utilized by prior studies to detect HTTP DDoS attacks. The review output reveals that signature-based detection methods are effective for known attack patterns, while anomaly-based detection excels at detecting previously unseen behaviors. However, the signature-based detection struggles to recognize new attack patterns, unlike anomaly-based detection. Both of these detections also experience significant challenges in differentiating between authentic users and automated attack tools when public proxies are used. This review concludes that signature-based and anomaly-based detection techniques remain inadequate for detecting the attack. This review also suggests that future research should focus on a hybrid detection to detect request headers in real-time and the multi-version HTTP protocol to improve detection accuracy.

Keywords: DDoS, DoS, Signature Detection, Anomaly Detection

** Corresponding author. abdulghafar@utm.my*

1. Introduction

Distributed Denial of Service (DDoS) consists of two main categories, known as network and application layer attacks. The application layer attack, known explicitly as HTTP DDoS, represents a critical threat to the availability and reliability of online services because it directly exploits the application layer where web servers deliver content to users. As noted by contemporary studies, the significance of this threat lies in its ability to mimic legitimate traffic patterns. Consequently, detection of the attack becomes challenging. Therefore, the growing sophistication of HTTP DDoS attacks demands urgent academic and practical attention.

At the technical level, HTTP DDoS attacks operate through multiple vectors, such as forged request headers, automated attack tools, and the misuse of public proxies. More importantly, the existence of these vectors demonstrates that adversaries can disguise malicious queries as authentic requests. This situation complicates the distinction between benign and harmful traffic. For example, automated tools such as LOIC, HOIC, and SlowHTTPTest generate traffic that is equal to user-initiated browsing, while proxies further obscure the origin of the connection. As a result, conventional detection techniques, particularly signature-based methods, encounter severe limitations in accurately identifying the attack. In addition, the challenge is magnified when encryption protocols, such as HTTPS, conceal traffic details. Although encryption secures communication channels, it simultaneously provides attackers with a mechanism to deliver malicious payloads undetected. Furthermore, the evolving HTTP protocol versions, such as HTTP/2 and HTTP/3, introduce new attack surfaces, enabling multiplexed and asymmetric assaults that overwhelm servers with minimal resources. Thus, the interplay between legitimate user behavior and adversarial manipulation highlights the need to develop detection techniques that can operate across various versions, encryption schemes, and traffic channels.

This study aims to review existing solutions based on signature-based and anomaly-based detection for identifying HTTP DDoS attacks and provide recommendations for future directions to effectively detect the attack. The paper comprises several sections, with Section 2 serving as the literature review, which is divided into subsections numbered 2.1 to 2.8. The discussion and future direction are covered in Section 3, while Section 4 covers the conclusion of this review.

2. Literature Review

2.1 Authentic and False Request Headers

HTTP traffic contains request and response messages [1]. The process of a user browsing online content through a web browser is referred to as an HTTP request, while the web server's response to these requests is known as an HTTP response. The HTTP requests contain request headers generated automatically and will end

with a Carriage Return Line Feed (CRLF), which is the line break for each request header. The CR refers to \r, and LF refers to \n, indicating the line break of each request header. The existence of \r\n\r\n demonstrates that the headers are complete [2]. The last request headers in genuine HTTP requests contain two CRLFs. However, during the execution of an HTTP DDoS attack, only a single CRLF exists at the end of the last request header [3]. The existence of a single CRLF increases the web server's waiting time for an HTTP request to complete the transaction. Due to this situation, server resource consumption can significantly increase, leading to a web server becoming unresponsive and crashing [4]. Yun, Xie [5] explain that request headers are vulnerable components because anyone, including adversaries, can modify them to imitate benign HTTP request traffic. Figure 1 illustrates the genuine request headers, which include two CRLFs and a single CRLF, generated by the HTTP DDoS attack.

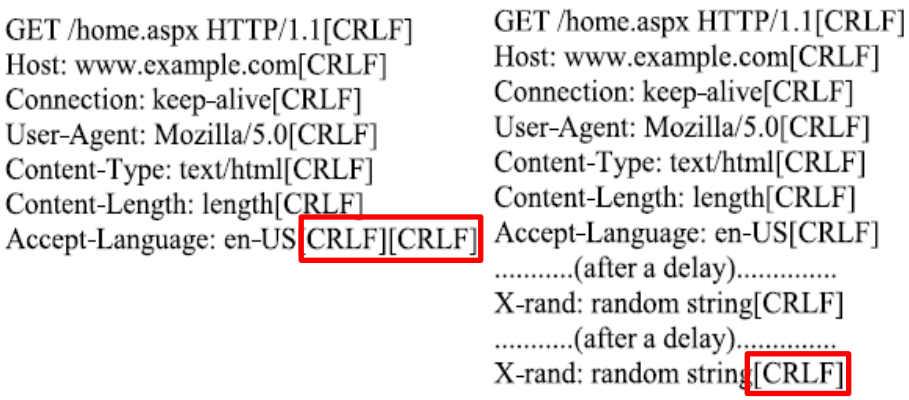

Figure 1: Genuine and False Request Headers

The authentic CRLF in the final request headers can also be represented as binary code, such as 0d 0a 0d 0a (CRLF, CRLF) (Nithyanandam & Dhanapal, 2019). Nevertheless, when the HTTP DDoS attack was executed, it was found that only 0d 0a (CRLF) was present in the request headers. Figure 2 demonstrates the binary code for the HTTP DDoS attack.

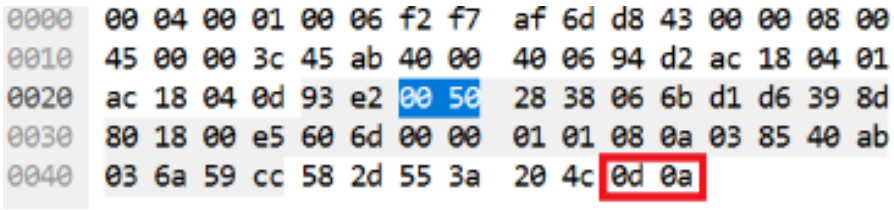

Figure 2: Forged Binary Code

2.2 Attack Channels via Automation and Proxies

Automated attack tools enable hackers to specify a set of URLs for launching HTTP DDoS attacks. [6]. Automated attack tools can generate random authentic request headers, making it difficult to recognize the attack pattern. Dhanapal and Nithyanandam [7] observed that request headers formed by hackers comprised genuine user agents, incorrect URLs, and repeated URLs. The existence of automated attack tools like HOIC, LOIC, Golden Eye, Hulk, DDOSIM, Rudy, and SlowHTTPTest [4, 7-9], which are shared through open channels like GitHub and specific online communities in the dark web, allows hackers to keep on changing and enhancing the code to be more sophisticated. Sangodoyin, Akinsolu [9] identified that the existence of sophisticated tools, the continuous modification of attack toolkits, and the development of automated attack tools contribute to the difficulty of detecting attacks [10].

Aside from the scenario explained above, hackers can exploit the widespread use of public proxies by misusing the service to launch HTTP DDoS attacks. In this circumstance, hackers can execute automated attack tools from a single machine to instruct public proxies to generate traffic aimed at the victim's web server. Consequently, a substantial amount of HTTP requests will be generated through public proxies, and the Server will collapse due to being overloaded. Due to proxies mediating between users and web servers, identifying authentic users and HTTP DDoS attacks while accessing and attacking a web server is complicated. To utilize a proxy, users must successfully establish a Transmission Control Protocol (TCP) connection. Once done, the proxy will route the user request to a web server for processing. Figure 3 illustrates the TCP handshake process between the client and proxy, highlighting the challenge in differentiating between legitimate users and HTTP DDoS attacks. This complexity arises from the fact that such attacks can be initiated by either the user or the hackers who organized the attack.

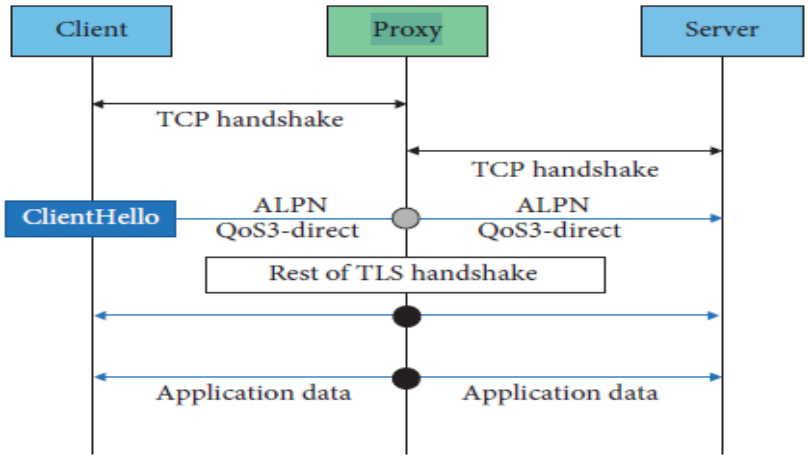

Figure 3: Proxy Handshake between Client and Web Server Al-Dailami, Ruan [11]

2.3 HTTP Version

Certain web servers still utilize HTTP 1.1 due to current technology constraints. Chatzoglou, Kouliaridis [12] highlighted that a few web servers, notably IIS 10, had trouble with HTTP version 3. As a result, they switched to HTTP version 2 or HTTP version 1.1 for client-server communication. HTTP versions 2 and 3 can perform better than HTTP version 1.1. HTTP version 2 utilizes a compression method known as HPACK to reduce the size of data, improve web performance, and eliminate redundant information in HTTP header packets. On the other hand, this benefit significantly increased the web server's workload, as compressed headers must be decompressed whenever HTTP/2.0 connects to HTTP/1.1 for data exchange. Since HPACK compression allows users to generate more HTTP requests using the same bandwidth, malicious actors can immediately overwhelm a web server. HTTP version 2 introduces a new DDoS attack vector called a Multiplexed Asymmetric attack, which can overwhelm web server resources by utilizing limited machines to generate the attack traffic [13]. HTTP DDoS attacks launch over various HTTP protocol versions. However, the current proposed detection method is limited to HTTP 1.1. It is time for researchers to develop detection algorithms capable of inspecting HTTP requests across all versions. Today's hackers are highly skilled and well-equipped with various automated tools, enabling them to launch successful attacks against businesses and governments [14, 15]. NSFOCUS [16] highlights that DDoS mitigation techniques must evolve in response to the emergence of new attack vectors. Figure 4 demonstrates the distribution of HTTP versions.

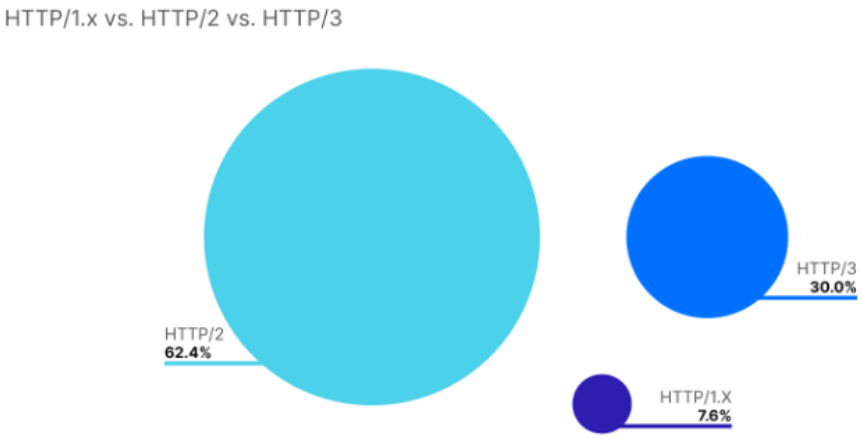

Figure 4: Distribution of HTTP versions [17]

2.4 HTTP Traffic Encryption

Encryption has become the standard for most Internet traffic due to growing concerns about privacy and security. When it comes to encrypting conversations, HTTPS will naturally employ the Transport Layer Security (TLS) protocol. TLS is a widely used protocol for ensuring the privacy and integrity of data transmitted

across TCP connections. According to Moura, Lopes [18], the protocol enables server/client applications to communicate securely over a channel, thereby preventing eavesdropping, tampering, and message forgery. TLS has emerged as the predominant end-to-end encryption standard for transmitting web information [11]. This protocol first appeared under the name *Secure Sockets Layer* (SSL), and the final 'S' in HTTPS stands for 'Secure', demonstrating in the web browser that the communication is secure [18]. To establish a secure connection, both parties must initiate a TCP handshake, exchanging the required information to construct an encrypted channel and encrypt data during transmission [19].

SSL/TLS completely encrypts the data during transmission, leaving the data, including request headers, vulnerable to attacks. Xiao, Zhang [20] stated that governments, enterprises, and banks have begun to deploy HTTP over SSL/TLS to protect the security of sensitive data during transmission. As a result, hackers construct forged request headers that appear identical to authentic ones to launch HTTP DDoS attacks over encrypted channels, due to the lack of a verification method for the legitimacy of these headers. Zhao, Peng [21] highlight that the complexity of identifying HTTP DDoS attacks stems from the fact that the attack utilizes authentic request headers. As a result, it is tricky to differentiate the authenticity of the traffic in the attack [22, 23]. Figure 5 illustrates the communication between the client and Server in establishing a secure channel, indicating that the data is only encrypted during transfer.

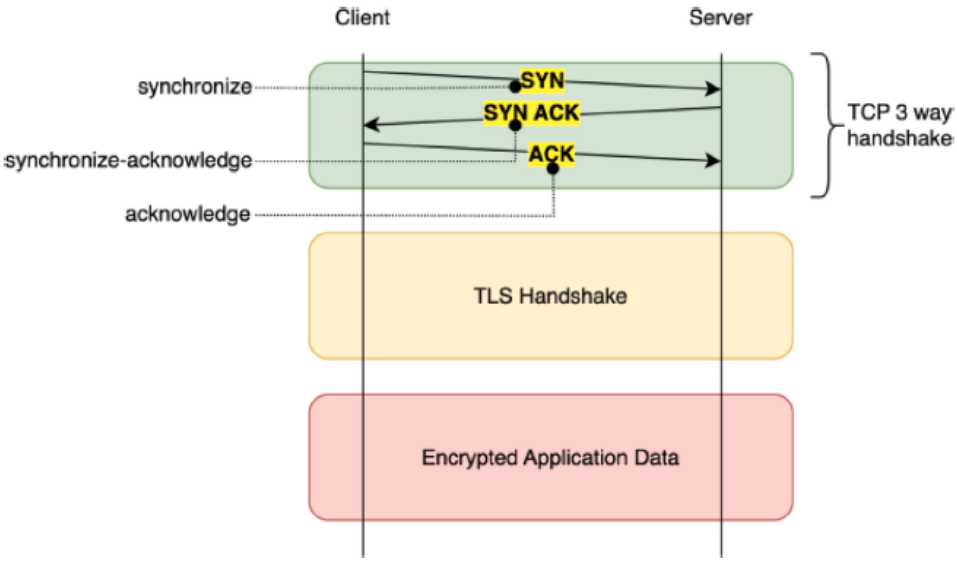

Figure 5: Secure Channel Communication Between Clients and Server [19]

Data transmitted over SSL/TLS is encrypted and protected to ensure that communications between the client and Server are not intercepted by hackers and modified during transmission [20]. However, Di Martino, Quax [24] emphasized that the protocol is vulnerable to a Man-in-the-Middle (MitM) attack if a client does

not verify the security certificate during the TLS handshake, as hackers have the ability to create a certificate that is similar to the original one, enabling them to establish a connection with a server and decode the transmitted data. According to Alashwali, Szalachowski [25], hackers can eavesdrop passively, inject malicious traffic, modify, drop, replay, or redirect messages in an HTTPS channel. Due to the aforementioned factor, hackers execute the MitM attack as the initial steps to observe authentic request headers and construct the same headers to launch HTTP DDoS attacks. Consequently, the attack traffic appears genuine and is difficult to differentiate because it generates forged request headers identical to the authentic version. A potential approach to resolving the mentioned issue is to encrypt the request headers before transmission, thereby mitigating the risk of request header forgery to prevent the attack from replicating the headers. Figure 6 demonstrates the activity done by hackers to launch attack over HTTPS channel.

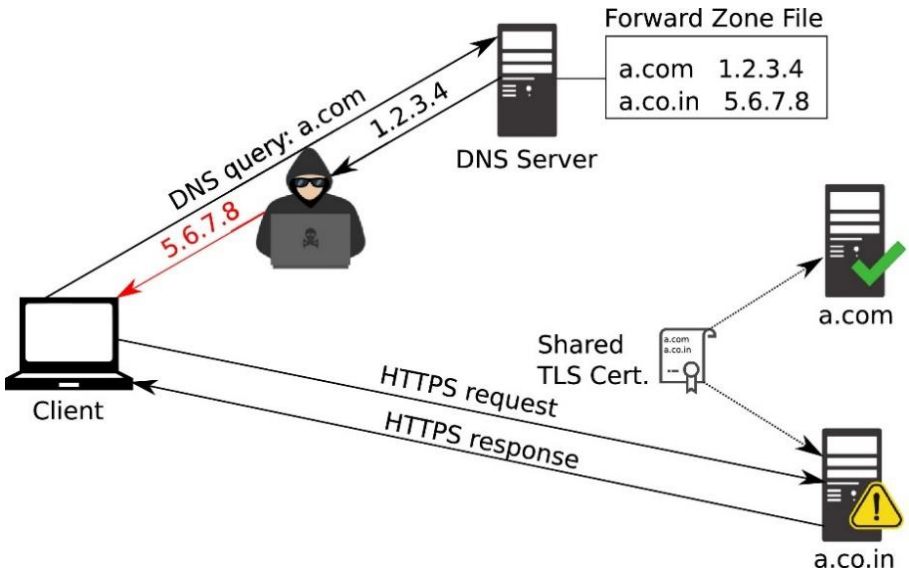

Figure 6: Activity Done by Hackers to Sniff Traffic in HTTPS Channel

2.5 Related Work Based on Signature-Based Detection Algorithm

The signature-based approach relies on network behavior and is also referred to as misuse, knowledge-based, rule-based, and pattern-based detection [26]. The type of detection relies on knowledge-based databases and a defined set of policies for implementation. Numerous studies have employed signature-based detection for the rapid identification of HTTP DDoS attacks. For instance, Praseed and Thilagam [6] utilize HTTP request headers to develop an Early Detection Module (EDM) for detecting DDoS attacks. The proposed work supervises incoming requests to verify that the source connection is non-repetitive and contrasts the traffic with the

signature database to determine the traffic status. An alternative method for identifying the attack involves redirecting internet traffic to identify the hacker, as proposed by Gonçalves et al. (2022).

Mohammadi, Lal [4] proposed a detection that was implemented as a defense module on Software-Defined Networking (SDN) to recognize HTTP DDoS attacks. SHFD extracts the source IP address from the SYN packet and saves it onto a list known as *HostReqList.* The incoming traffic will be detected as suspicious if the source IP address fails to deliver complete request headers within a specified time window. Park, Kim [27] also deploys SDN to detect HTTP DDoS, categorized as flooding attacks (SDN). The scholar uses a web server to inspect incoming traffic, forwarding it to the SDN controller and blocking the connection if the incoming IP address is suspicious.

Another idea for detecting the attack is to use page separation and resource allocation strategies introduced by Patidar and Somani [28]. It has a function called allowlist, where requests from this source are deemed authentic because traffic from this section uses an authentication channel once credentials are provided. Rao Varre and Bayana [29] suggested using a request rate monitor to control the amount of incoming traffic and analyze authentic and malicious data. Every request rate is monitored in accordance with the server capacity allocated for each request. The traffic monitoring and rate limit concept is applied not only by Rao Varre and Bayana [29] but also by Zhao and, who introduced the concept of behavioral utility to portray the network pattern. The study suggests that the network status change caused by the attack can be characterized as attack behavior. The studies utilize load balancing and rate limits to control the maximum number of connections across different attack scales, and the exact mechanism is employed to determine the threshold status.

2.6 Related Work Based on AI Machine Learning Detection Algorithm

Artificial Intelligence (AI) and Machine Learning (ML) have become known as crucial technologies in cybersecurity, particularly in the detection of HTTP DDoS attacks. Artificial Intelligence and Machine Learning offer extensive capabilities by analyzing vast amounts of network data, identifying patterns, and distinguishing between normal and anomalous behaviors. Sarker, Furhad [30] stated that artificial intelligence utilizes machine learning models, including traditional models such as Naive Bayes and decision trees. Al-gethami and Aljuhani [31] state that diverse machine learning models, such as K-Nearest Neighbors (KNN) and Random Forest, can be utilized to identify HTTP DDoS attacks. These models are often used to analyze network traffic patterns and identify anomalies. Research conducted by Alghazzawi, Bamasag [32] demonstrated that combining various machine learning

models, such as Convolutional Neural Networks (CNN) and Bidirectional Long Short-Term Memory (BILSTM), can substantially enhance detection precision.

Mohammadi, Lal [33] propose Software Defined Network (SDN) and machine learning algorithms to detect and mitigate HTTP flooding attacks. They utilize an SDN controller to monitor the HTTP connections and document each connection in the log table. All HTTP flows at the end of a time window are inspected to detect the presence of the attack. Feng, Li [34] suggests that a reinforcement-learning-based model can detect and mitigate HTTP DDoS attacks. The module is designed to train various metrics related to server load, dynamic user behavior, and the victim network load. A similar detection matrix was suggested by Praseed and Thilagam [23] using a two-phase detection approach that incorporates server logs and access traces to capture the behavioral dynamics of legitimate users in the form of annotated Probabilistic Timed Automata (PTA). The detection phase is designed to intercept and examine all incoming connections, detecting traffic patterns that do not conform to the learned model. Another equivalent technique introduced by Abubakar, Aldegheishem [35] adopted traffic behavior analysis, packet header validation, protocol validation, and traffic matching with datasets.

Yu, Yu [36] propose a semi-supervised machine learning approach that combines spectral clustering and random forest. The study utilized a dataset containing seven discrete feature variables: guest_login, logged_in, land, flag, is_host_login, service, and protocol_type. Beitollahi, Sharif [10] also analyzed variables available in the dataset, utilizing a Genetic Algorithm (GA) for data gathering, cleaning, normalization, and feature selection. The proposed detection method distinguishes malicious requests based on features available in the dataset, such as protocol type, the percentage of connections with SYN errors, and the percentage of connections that use the same and different services. Table 1 indicates the summary of the proposed detection by prior studies.

Table 1: Detection Algorithm Proposed by Prior Studies

No.	Reference	Dataset/Automated Tools	Method Used	Technique
1.	Praseed and Thilagam [6]	SDSC-HTTP CLARKNET-HTTP	HTTP request patterns Sample Entropy	Signatured-Based
2.	Mohammadi, Lal [4]	Slow HTTP Test	Entropy And Hellinger Distance	Signatured-Based
3.	Rao Varre and Bayana [29]	Botnet	Resource Request Rate Monitor Release Request for Process Invisible Challenge	Signatured-Based
4.	Gonçalves, Couto [37]	Bonesi	Server Redirection	Signatured-Based
5.	Park, Kim [27]	HTTP DDoS flooding	CAPTCHA	Signatured-Based
6.	Patidar and Somani [28]	Generate traffic via Ubuntu OS	Page separation Resource allocation	Signatured-Based
7.	Zhao, Peng [21]	Burp Suite LOIC JMeter	Network throughput rate TCP data segment transmission rate IP datagram transmission rate Transaction failure rate Average traffic arrival time Server CPU utilization	Signatured-Based
8.	Beitollahi, Sharif [10]	NSL-KDD	Radial Basis Function Cuckoo Search Algorithm Genetic Algorithm	Anomaly-Based
9.	Mohammadi, Lal [33]	Slow HTTP Test	KNN Naive Bayes Decision Tree Random Forest	Anomaly-Based
10.	Praseed and Thilagam [23]	SDSC-HTTP CLARKNET-HTTP	Probabilistic Timed Automata (PTA) Suspicion Scoring Mechanism	Anomaly-Based
11.	Yu, Yu [36]	NSL-KDD	Semi-Supervised Learning	Anomaly-Based

* Corresponding author. abdulghafar@utm.my

No.	Reference	Dataset/Automated Tools	Method Used	Technique
			• Spectral Clustering • Random Forest	
12.	Abubakar, Aldegheishem [35]	• TCP Replay • LOIC • JMeter	• Traffic Behaviour Analysis • Packet Header Validation • Protocol Validation • Traffic Matching	Anomaly-Based
13.	Feng, Li [34]	• Slowloris • HULK • DDos Simulator	• Behavioural Information • Network Load • System Load • Reinforcement Learning	Anomaly-Based

2.7 Complexity Detecting HTTP DDoS Through Public Proxies

Identifying HTTP DDoS attacks becomes more complex when public proxy providers use various proxy header names to indicate the origin of traffic from proxies. The inconsistency in the public proxy header's name complicates the detection of the attack, as the Server, the receiver, cannot determine whether the source connection originates from a proxy or a direct connection (without proxies). Another circumstance that produces complexity is that the web servers are specifically programmed not to recognize the source connection, whether from a web browser or automated attack tools. According to Díaz-Verdejo, Estepa Alonso [22], distinguishing authentic HTTP requests is extremely challenging due to the dynamic and robust structure of web servers. For this reason, the method used by prior studies, which employs signature-based detection, is ineffective in detecting the attack. For instance, the proposed solution by Park, Kim [27] increases server workload because it uses the web server to inspect incoming traffic and forward it to the SDN controller, which blocks the connection if the incoming IP address is suspicious. Chatzoglou, Kouliaridis [12] noted that significant incoming traffic can lead to CPU exhaustion on the Server, while Patidar and Somani (2021) observed that a genuine user requires 6,000–7,000 milliseconds per request. As a consequence of this situation, the time required for HTTP DDoS attacks increases when the Server receives a large number of requests, leading to longer response times and potential timeouts.

Patidar and Somani [28] suggest strategies for page separation and resource allocation. Compared to others, the primary advantage of this method is that it can distribute traffic across multiple servers, thereby reducing the workload. However, the proposed solution cannot categorize traffic from different channels, particularly from proxies, because their scattered locations make them challenging to identify. As a result, allocating resources could be inaccurate. Aside from that, resource allocation strategies face a challenge in handling large amounts of traffic because public proxies can generate substantial traffic on a large scale. Consequently, this further complicates the identification process, as detection devices must scan every bit of traffic in a vast amount in sequential order. Mohammed Sharif and Beitollahi [38] explained that HTTP DDoS attacks specifically target application resources with high request rates.

Rao Varre and Bayana [29] and Zhao, Peng [21] utilize a request rate monitor to detect the attack. Although the method has the benefit of observing a high request rate, it presents a significant drawback. Legitimate users are unable to access web content because authentic users and malicious actors utilize the same public IP Address, resulting in a higher request rate and a blocked connection. Another study with the same limitation is Praseed and Thilagam [6], as the proposed work detects a malicious request if the source connection repeatedly generates traffic against the web server at a higher rate. HTTP DDoS attacks through public proxies are identical to the flash crowd attack, as they appear to be legitimate traffic. Rizvi, Mirkovic [39] noted that the attack employs authentic traffic from compromised hosts. The proposed work by Mohammadi, Lal [4] highlights a limitation in identifying the source connection, whether it is from public proxies or vice versa. Despite Sree and

* Corresponding author. abdulghafar@utm.my

Bhanu [40] stating that request headers could be used to detect attacks because the access log contains headers, the request headers generated by a web browser differed from those generated by public proxies.

The machine learning model also has drawbacks that require further attention, as it relies on a training model where the dataset serves as the primary source of the model [41]. The proposed solution by Beitollahi, Sharif [10], Praseed and Thilagam [23], Yu, Yu [36] is unable to detect recent attack patterns such as HTTP DDoS originating through public proxies due to the use of outdated datasets. Díaz-Verdejo, Estepa Alonso [22] state that the attack in the existing dataset is limited and outdated.

Feng, Li [34] employ machine learning methods that utilize tools such as Slowloris, HULK, and DDoS Simulator, rather than relying solely on datasets. Even though these studies employed various methods related to server load, dynamic user behavior, and victim network load, they are still incapable of detecting HTTP DDoS launches over proxies, as the tools cannot generate HTTP DDoS traffic through them. Mohammadi, Lal [33] extends their work from signature-based detection to anomaly detection but still struggles to distinguish between proxies used solely to access the web server's content and those employed by malicious actors to overload the web server with fake traffic. This circumstance, due to limitations in using detection attributes such as source and destination IP addresses and the port number, is insufficient for detecting HTTP DDoS execution over proxies.

2.8 Limitations of Signature and Anomaly Detection

After rigorous examination, it was discovered that signature-based detection [4, 6, 21, 27-29, 37] and anomaly detection [10, 23, 33-36] experience complexity in detecting forged request headers, as the headers can be created through automated attack tools and appear genuine once executed and delivered to a web server. The proposed solution by Rao Varre and Bayana [29] and Park, Kim [27] almost addresses the issue of identifying the source connection. Nevertheless, it relies on human interaction, which complicates user access. This could potentially lead to inaccurate responses from users, making it unsuitable for all levels of users, especially the elderly or those with limited computer skills.

The proposed work by Mohammadi, Lal [4] can also partially detect the source identity, which may originate from automated attack tools or a web browser, as it identifies incoming connections as HTTP DDoS attacks if the source IP address delivers incomplete request headers. However, the proposed work does not consider detecting the use of automated attack tools that hackers can use to construct large-scale attack volumes. Due to this, the detection is unable to distinguish the platform used by the source connection. Bialczak and Mazurczyk [42] emphasized that applications or programs beyond web browsers can utilize the HTTP protocol. For this reason, hackers employ automated attack tools that follow the same protocol to generate a significant amount of traffic, thereby overwhelming a web server. Given

these circumstances, it is essential to differentiate between automated attack tools and web browsers to quickly recognize the attack before the web server is overwhelmed by a significant number of connections made by hackers.

Automated tools, web browsers, and forged request headers are interconnected, as hackers can use automated attack tools to mimic the request headers constructed by web browsers, thereby executing HTTP DDoS attacks. Consequently, the detecting devices become confused and allow entry to a web server. Additionally, hackers use automated attack tools to construct forged request queries, employing ASCII characters similar to those generated by legitimate users through web browsers. Figure 7 illustrates the code structure of automated attack tools using these characters, while Table 2 lists the ASCII characters.

```
def buildblock(size):
        out_str = ''
        LowerCase =range(97,122)
        UpperCase =range(65,90)
        Numeric =range(48,57)
        SpecialChar =range(33,47)
        CombineString = LowerCase + UpperCase + Numeric + SpecialChar

        for i in range(0, size):
                a = random.choice(CombineString)
                out_str += chr(a)
        return(out_str)
```

Figure 7: Automated Attack Tools Code with ASCII Characters

Table 2: ASCII Character

Type	ASCII Character	Symbol Generate
Lower Case	97 and 122	a-z
Upper Case	65 and 90	A-Z
Numeric	48 and 57	0-9
Special character	33 and 57	!"#$%&`()*+'-./

Based on the limitations elaborated in Sections 2.7 and 2.8, a gap that needs to be addressed is apparent. Due to this circumstance, detecting HTTP DDoS attacks executed over public proxies and distinguishing forged request headers is essential for future research. Aside from that, it is also vital to differentiate between legitimate web browsers and automated attack tools when accessing and attacking web servers in real time. As a result, early detection can be achieved before the attack crashes a web server with a substantial number of connections.

3. Discussion and Future Direction

The difficulty in detecting HTTP DDoS attacks stems from the fact that the forged request headers generated by the attack are identical to authentic request

headers, enabling them to evade detection. Moreover, the attack originated from various points, including automated attack tools and public proxies, complicating its identification. Although various methods have been applied, research into detecting HTTP DDoS executed over public proxies and identifying forged request headers remains a significant research gap. Another critical issue that requires serious attention is the inability of recent detection methods to differentiate between genuine web browsers and automated attack tools when accessing and attacking web servers. The existence of this problem has a detrimental impact, as a web server is forced to process both legitimate and illegitimate HTTP requests concurrently. As a result, the web server receives a heavy workload and crashes.

Proxies function as intermediaries between users and web servers, enabling individuals to gain a high-speed connection and browse online content hosted by a server. However, hackers can connect to public proxy servers to conceal their IP addresses and execute HTTP DDoS attacks. As a result, it provides a layer of anonymity, making it difficult for defenders and authorities to identify the originality of the traffic because genuine users and hackers use the proxy simultaneously. Hackers may utilize multiple public proxies in rotation, switching between them during attacks, which can complicate the ability of security devices to block or consistently detect malicious traffic. Furthermore, the public proxies are scattered, the services are free, and no authentication or subscription is required, making this platform the first choice for launching an attack. HTTP DDoS attacks conducted through public proxies are challenging to differentiate due to the similarity in internet traffic, as users often utilize public proxies to access online content, while hackers exploit them to launch HTTP DDoS attacks. This increases the need for future studies to discover an effective solution to address these issues.

Another problem that requires serious attention by future studies is the request header forgery, which is executed through HTTP DDoS. The request headers contain information about the request or the client making the HTTP request. It provides essential metadata for the Server to process the request appropriately. The headers typically include details such as the type of browser or client, the type of data the client can accept, and information about the content being sent. However, hackers can modify the headers, making the attack traffic appear authentic and difficult to distinguish from legitimate traffic. This problem has been partially addressed by Mohammadi, Lal [4], who proposed a solution to detect HTTP DDoS attacks by measuring the number of headers generated by the source requester. However, incomplete headers are not the only patterns produced by the attack. Praseed and Thilagam [6] highlighted that the attack is strikingly similar to legitimate traffic and difficult to detect.

According to Park, Kim [27], the attack will overwhelm a web server because of the significant number of HTTP requests. Differentiating between automated attack tools and genuine web browsers is a complex task. Web servers are designed to ignore the platform used by the requester, which can be either automated attack tools or web browsers, making the web server vulnerable to HTTP DDoS attacks. Thus, identifying the platform from which the source connection originates requires

further exploration to detect the attack early, before the web server is overloaded by the substantial traffic it generates.

4. Conclusion

This study assessed the effectiveness of signature-based and anomaly-based methods in detecting HTTP DDoS attacks. Signature-based detection can promptly identify attack patterns; however, it lacks the ability to react to emerging threats. Anomaly-based detection can identify novel behaviors; nevertheless, it frequently produces false positives, leaving it less trustworthy. Identifying the use of forged request headers, encrypted HTTPS traffic, automated attack tools, and public proxies is challenging. These malicious tactics generate attack traffic that mimics legitimate requests, causing servers to exceed their limits and ultimately leaving them unable to respond due to the high load they receive.

Ultimately, the synthesis of prior studies indicates that addressing the limitations of existing methods requires innovative, layered defenses. Therefore, future research should prioritize the development of hybridized, intelligent detection systems that can withstand adversarial manipulation while sustaining the resilience of critical web infrastructures. This study thus contributes by clarifying the gaps in current approaches and by charting a path toward more reliable defenses against HTTP DDoS attacks.

Acknowledgments:

This research took place and received financial support from Universiti Teknologi Malaysia, UTMER, PY/2024/00571 / Q.K130000.3857.42J38

Conflict of Interest

The author declares that there is no conflict of interest regarding the publication of this paper.

References:

[1] Almutairi, S., et al., *Hybrid Botnet Detection Based on Host and Network Analysis.* Journal of Computer Networks and Communications, 2020. 2020: p. 1-16.

[2] Bhardwaj, A., et al., *Distributed denial of service attacks in cloud: State-of-the-art of scientific and commercial solutions.* Computer Science Review, 2021. 39: p. 100332.

[3] Nithyanandam, P. and A. Dhanapal, *The Slow HTTP Distributed Denial of Service Attack Detection in Cloud.* Scalable Computing: Practice and Experience, 2019. 20(2): p. 285-298.

[4] Mohammadi, R., et al., *Software defined network-based HTTP flooding attack defender.* Computers and Electrical Engineering, 2022. 101: p. 108019.

[5] Yun, X., et al., *Detecting unknown HTTP-based malicious communication behavior via generated adversarial flows and hierarchical traffic features.* Computers & Security, 2022. 121.

[6] Praseed, A. and P.S. Thilagam, *HTTP request pattern based signatures for early application layer DDoS detection: A firewall agnostic approach.* Journal of Information Security and Applications, 2022. 65.

[7] Dhanapal, A. and P. Nithyanandam, *An OpenStack Based Cloud Testbed Framework for Evaluating HTTP Flooding Attacks.* Wireless Networks, 2019.

[8] Hosseini, S. and M. Azizi, *The Hybrid Technique for DDoS Detection with Supervised Learning Algorithms.* Computer Networks, 2019. 158: p. 35-45.

[9] Sangodoyin, A.O., et al., *Detection and Classification of DDoS Flooding Attacks on Software-Defined Networks: A Case Study for the Application of Machine Learning.* IEEE Access, 2021. 9: p. 122495-122508.

[10] Beitollahi, H., D.M. Sharif, and M. Fazeli, *Application Layer DDoS Attack Detection Using Cuckoo Search Algorithm-Trained Radial Basis Function.* IEEE Access, 2022. 10: p. 63844-63854.

[11] Al-Dailami, A., et al., *Qos3: Secure caching in https based on fine-grained trust delegation.* Security and Communication Networks, 2019. 2019: p. 1-16.

[12] Chatzoglou, E., et al., *A hands-on gaze on HTTP/3 security through the lens of HTTP/2 and a public dataset.* Computers & Security, 2023. 125: p. 103051.

[13] Praseed, A. and P.S. Thilagam, *Multiplexed Asymmetric Attacks: Next-Generation DDoS on HTTP/2 Servers.* IEEE Transactions on Information Forensics and Security, 2020. 15: p. 1790-1800.

[14] Mustafa, I., et al., *A lightweight post-quantum lattice-based RSA for secure communications.* IEEE Access, 2020. 8: p. 99273-99285.

[15] Ahmed, S., et al., *Effective and Efficient DDoS Attack Detection Using Deep Learning Algorithm, Multi-Layer Perceptron.* Future Internet, 2023. 15(2): p. 76.

[16] NSFOCUS, *DDoS Attack Lanscape*. 2020.

[17] Omer Yoachimik and J. Pacheco. *DDoS Threat Report For 2023 Q3*. 2023 [cited 2023 19 November 2023]; Available from: https://blog.cloudflare.com/ddos-threat-report-2023-q3/.

[18] Moura, R., et al., *MultiTLS: using multiple and diverse ciphers for stronger secure channels.* Computers & Security, 2023: p. 103342.

[19] Abdelhafez, M., S. Ramadass, and M. Abdelwahab, *TLS Guard for TLS 1.3 Zero Round-Trip Time (0-RTT) in a Distributed Environment.* Journal of King Saud University-Computer and Information Sciences, 2023: p. 101797.

[20] Xiao, C., et al., *Energy-efficient crypto acceleration with HW/SW co-design for HTTPS.* Future Generation Computer Systems, 2019. 96: p. 336-347.

[21] Zhao, X., et al., *Defending Application Layer DDoS Attacks via Multidimensional Parallelotope.* Security and Communication Networks, 2020. 2020: p. 6679304.

[22] Díaz-Verdejo, J.E., et al., *A critical review of the techniques used for anomaly detection of HTTP-based attacks: taxonomy, limitations and open challenges.* Computers & Security, 2023. 124.

[23] Praseed, A. and P.S. Thilagam, *Modelling Behavioural Dynamics for Asymmetric Application Layer DDoS Detection.* IEEE Transactions on Information Forensics and Security, 2021. 16: p. 617-626.

[24] Di Martino, M., P. Quax, and W. Lamotte, *Knocking on ips: Identifying https websites for zero-rated traffic.* Security and Communication Networks, 2020. 2020: p. 1-14.

[25] Alashwali, E.S., P. Szalachowski, and A. Martin, *Exploring HTTPS security inconsistencies: A cross-regional perspective.* Computers & Security, 2020. 97: p. 101975.
[26] Myint Oo, M., et al., *Advanced Support Vector Machine- (ASVM-) Based Detection for Distributed Denial of Service (DDoS) Attack on Software Defined Networking (SDN).* Journal of Computer Networks and Communications, 2019: p. 1-12.
[27] Park, S., et al., *HTTP DDoS Flooding Attack Mitigation in Software-Defined Networking.* IEICE Transactions on Information and Systems, 2021. E104.D(9): p. 1496-1499.
[28] Patidar, A. and G. Somani, *Serving while attacked: DDoS attack effect minimization using page separation and container allocation strategy.* Journal of Information Security and Applications, 2021. 59.
[29] Rao Varre, D.N.M. and J. Bayana, *A Secured Botnet Prevention Mechanism for HTTP Flooding Based DDoS Attack*, in *2022 3rd International Conference for Emerging Technology (INCET)*. 2022. p. 1-5.
[30] Sarker, I.H., M.H. Furhad, and R. Nowrozy, *Ai-driven cybersecurity: an overview, security intelligence modeling and research directions.* SN Computer Science, 2021. 2(3): p. 173.
[31] Al-gethami, W. and A. Aljuhani. *Detection of http attacks using machine learning*. in *2022 2nd International Conference on Computing and Information Technology (ICCIT)*. 2022. IEEE.
[32] Alghazzawi, D., et al., *Efficient detection of DDoS attacks using a hybrid deep learning model with improved feature selection.* Applied Sciences, 2021. 11(24): p. 11634.
[33] Mohammadi, R., C. Lal, and M. Conti, *HTTPScout: A Machine Learning based Countermeasure for HTTP Flood Attacks in SDN.* International Journal of Information Security, 2022.
[34] Feng, Y., J. Li, and T. Nguyen. *Application-Layer DDoS Defense with Reinforcement Learning*. in *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*. 2020.
[35] Abubakar, R., et al., *An Effective Mechanism to Mitigate Real-Time DDoS Attack.* IEEE Access, 2020. 8: p. 126215-126227.
[36] Yu, X., et al., *WEB DDoS Attack Detection Method Based on Semisupervised Learning.* Security and Communication Networks, 2021. 2021: p. 1-10.
[37] Gonçalves, D.S.M., R.S. Couto, and M.G. Rubinstein, *A Protection System Against HTTP Flood Attacks Using Software Defined Networking.* Journal of Network and Systems Management, 2022. 31(1).
[38] Mohammed Sharif, D. and H. Beitollahi, *Detection of application-layer DDoS attacks using machine learning and genetic algorithms.* Computers & Security, 2023. 135: p. 103511.
[39] Rizvi, A.S.M., et al., *Defending Root DNS Servers against DDoS Using Layered Defenses (Extended).* Ad Hoc Networks, 2023. 151: p. 103259.
[40] Sree, T.R. and S.M.S. Bhanu, *Detection of HTTP Flooding Attacks in Cloud Using Fuzzy Bat Clustering.* Neural Computing and Applications, 2019. 32(13): p. 9603-9619.
[41] Alhijawi, B., et al., *A survey on DoS/DDoS mitigation techniques in SDNs: Classification, comparison, solutions, testing tools and datasets.* Computers and Electrical Engineering, 2022. 99: p. 107706.
[42] Bialczak, P. and W. Mazurczyk, *Characterizing Anomalies in Malware-Generated HTTP Traffic.* Security and Communication Networks, 2020. 2020: p. 1-26.