International Journal of Electrical and Computer Engineering (IJECE)
Vol. 13, No. 4, August 2023, pp. 4268~4279
ISSN: 2088-8708, DOI: 10.11591/ijece.v13i4.pp4268-4279

Hypertext transfer protocol performance analysis in traditional and software defined networks during Slowloris attack

Anusha A. Murthy, Prathima Mabel John, Rama Mohan Babu Kasturi Nagappasetty
Department of Information Science and Engineering, Dayananda Sagar College of Engineering, Visvesvaraya Technological University, Bengaluru, India

Article Info

Article history:

Received Apr 19, 2022
Revised Dec 7, 2022
Accepted Dec 11, 2022

Keywords:

Apache benchmark tool
H2load benchmark tool
HTTP/2
Monte Carlo model
Slowloris
Software defined network

ABSTRACT

The extensive use of the internet has resulted in novel technologies and protocol improvisation. Hypertext transfer protocol/1.1 (HTTP/1.1) is widely adapted on the internet. However, HTTP/2 is found to be more efficient over transport control protocol (TCP). The HTTP/2 protocol can withstand the payload overhead when compared to HTTP/1.1 by multiplexing multiple requests. However, both the protocols are highly susceptible to application-level denial of service (DoS) attacks. In this research, a slow-rate DoS attack called Slowloris is detected over Apache2 servers enabled with both versions of HTTP in traditional networks and software defined networks (SDN). Server metrics such as server connection time to the webpage, latency in receiving a response from the server, page load time, response-response gap, and inter-packet arrival time at the server are monitored to analyze attack activity. A Monte Carlo simulation is used to estimate threshold values for server connection time and latency for attack detection. This work is implemented in a lab environment using virtual machines, Ryu controller, zodiac FX OpenFlow switch and Apache2 servers. This study also highlights SDN's security benefits over traditional networks.

Corresponding Author:

Anusha A. Murthy
Department of Information Science and Engineering, Dayananda Sagar College of Engineering
Shavige Malleshwara Hills, Kumarswamy Layout, Bengaluru-560 111, India
Email: anushamurthy183@gmail.com

1. INTRODUCTION

Traditional computer networks are characterized by the conventional networking method of using dedicated switches and routers for directing network traffic. The control, data and management planes are coupled in each device leading to a lack of scalability, performance, programmability and security issues [1]. This has led the networking industry to move towards a highly programmable, virtual, scalable and more secure networking paradigm called software defined network (SDN). SDN provides greater flexibility in programming the network from a single point by decoupling the control and data planes. The control and data planes communicate using the OpenFlow protocol. The controller in the control plane serves as the master, dictating how the entire network operates. Controller is programmed using an application programming interface (API). The difference between traditional networks and SDN is shown in Figure 1.

The most popular and widely adopted application protocol over the internet is hypertext transfer protocol (HTTP). The HTTP/1.1 protocol can perform optimizations such as keep-alive (an indicator for connection operation) connections, chunked transfer encoding, byte-range requests, and request pipelining. A limited connection between client and server makes a series of requests to wait for one request to complete before it can fire it off. This condition is termed head-of-line (HoL) blocking in HTTP/1.1 and it is a major drawback of this protocol. This happens when several requests rise on the web servers. The protocol fails to respond even to small requests due to less resource volume size on the server-side. Hence it fails to use transport control protocol (TCP) effectively. Access to the world wide web (www) always suffers from less internet speed due to overloading on the servers with multiple requests from clients [2].

This issue found in HTTP/1.1 was solved by the next upgraded version, namely HTTP/2, by the process of parallel response and reply mechanism [3]. According to a survey that was conducted in the year 2017, it was found that around 16% to 30% of websites have been migrated to the HTTP/2 version. Popular browsers such as Chrome, Firefox, and Edge have already supported this new version [4]. HTTP/2 helps in providing higher communication speeds over client and server connections through multiple requests and responses with the support of the message multiplexing feature. This can be achieved by breaking HTTP/2 messages into independent units called headers, data, settings, and control frames. The protocol allows these independent units to be prioritized and reassembled into one message within a single TCP connection [5]. Hence, it can interactively send multiple requests from a client and deliver its requirements from the webserver based on the priority assigned. It helps in the processing of binary messages with the help of binary message framing. By using streams, messages, and frames in a binary framing mechanism, the data can be interchanged between client and server.

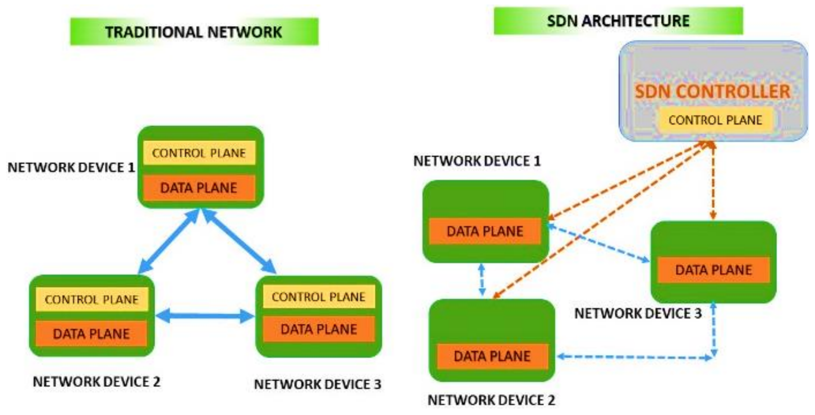
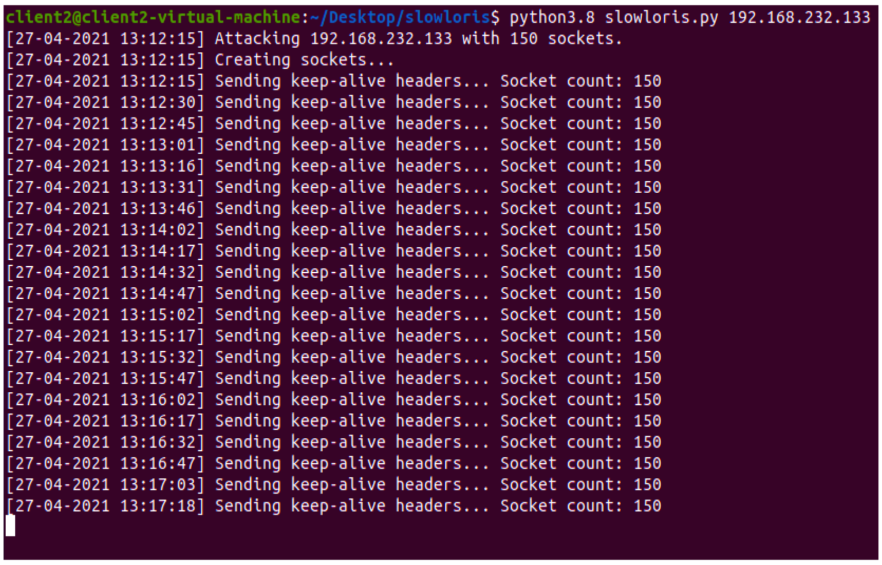


Figure 1. Difference between SDN and traditional network

In the case of HTTP/1.1, the authentication mechanism uses Windows Challenge/Response authentication, which provides relative security for data transfer, whereas in the case of HTTP/2, the security is enhanced by the addition of transport layer security (TLS) support over TCP connections. This provides a more secure channel for data transfer. With the multiplexing feature and server push, the web traffic in HTTP/2 will be reduced, hence the transmission of data is faster compared to the HTTP/1.1 protocol. The performance efficiency in HTTP/1.1 is reduced due to duplication of data across HTTP requests in multiple TCP connections. While the performance is enhanced by using HTTP/2 because of using prioritization tree to prioritize data packets with stream IDs, thereby consuming a lot of memory which will make the resources not available is another vulnerability of the protocol [6].

The vulnerability of these protocols will allow attackers to launch attacks that result in misuse and loss of data during message transmission. The common attacks over HTTP protocol are Garbage flood i.e., sending garbage binary data via HTTP packet, GET request flood i.e., sending huge data volume through GET request HTTP packet, Low and Slow denial of service (DoS) attacks i.e., sending low-rate/slow-rate HTTP packets thereby exhausting server resources [7]. Considering slow-rate DoS attacks over HTTP, one of the slow-rate DoS attacks called Slowloris is the main scope of this paper. Slowloris is a type of denial-of-service attack that allows an attacker to exhaust the resources of the targeted server by opening many simultaneous HTTP live connections between the attacker and the victim [8]. It is an application layer attack and falls under the category of low and slow rate DoS. This type of attack uses a minimum amount of bandwidth and tries to consume all server resources while providing a slow response to requests.

The targeted server creates many threads to handle concurrent open connections. Each thread attempts to stay active while waiting for the request to be completed. When the server's maximum connection limit is exceeded, it makes the server starve for resources and halts the present transmission of data between endpoints. The attack launched from the client-side on the server sends small HTTP requests continuously to keep the server thread alive, and this exhausts server resources for completing any other requests for webpage by other clients. The sample attack on the server IP is as shown in Figure 2.



```
client2@client2-virtual-machine:~/Desktop/slowloris$ python3.8 slowloris.py 192.168.232.133
[27-04-2021 13:12:15] Attacking 192.168.232.133 with 150 sockets.
[27-04-2021 13:12:15] Creating sockets...
[27-04-2021 13:12:15] Sending keep-alive headers... Socket count: 150
[27-04-2021 13:12:30] Sending keep-alive headers... Socket count: 150
[27-04-2021 13:12:45] Sending keep-alive headers... Socket count: 150
[27-04-2021 13:13:01] Sending keep-alive headers... Socket count: 150
[27-04-2021 13:13:16] Sending keep-alive headers... Socket count: 150
[27-04-2021 13:13:31] Sending keep-alive headers... Socket count: 150
[27-04-2021 13:13:46] Sending keep-alive headers... Socket count: 150
[27-04-2021 13:14:02] Sending keep-alive headers... Socket count: 150
[27-04-2021 13:14:17] Sending keep-alive headers... Socket count: 150
[27-04-2021 13:14:32] Sending keep-alive headers... Socket count: 150
[27-04-2021 13:14:47] Sending keep-alive headers... Socket count: 150
[27-04-2021 13:15:02] Sending keep-alive headers... Socket count: 150
[27-04-2021 13:15:17] Sending keep-alive headers... Socket count: 150
[27-04-2021 13:15:32] Sending keep-alive headers... Socket count: 150
[27-04-2021 13:15:47] Sending keep-alive headers... Socket count: 150
[27-04-2021 13:16:02] Sending keep-alive headers... Socket count: 150
[27-04-2021 13:16:17] Sending keep-alive headers... Socket count: 150
[27-04-2021 13:16:32] Sending keep-alive headers... Socket count: 150
[27-04-2021 13:16:47] Sending keep-alive headers... Socket count: 150
[27-04-2021 13:17:03] Sending keep-alive headers... Socket count: 150
[27-04-2021 13:17:18] Sending keep-alive headers... Socket count: 150
```

Figure 2. Slow-rate DoS-Slowloris attack on HTTP webserver

The behavior of the Slowloris attack on a traditional network and SDN having Apache2 servers enabled with HTTP/1.1 and HTTP/2 is discussed in this paper. The results of proposed detection mechanism are compared for both the networks. Furthermore, the key aspects of SDN architecture that contribute to better security than traditional networks are discussed.

In the work presented in [9], the concept of the HTTP/2 protocol along with its frame format, features, and vulnerabilities are explained. The author specifies efficient utilization of a single TCP connection with the concept of multiplexing in the new version. The investigation into HTTP/2 specifies the features of the protocol such as pipelining, response multiplexing, and server push and header compression that make it difficult for eavesdropping, thereby providing security against attack [10]. The paper focuses on the features of HTTP/2 protocol, i.e., multiplexing and push server which are misused to launch the distributed denial of service (DDoS) attack [11]. DDoS, which caused a disaster in HTTP/1.1, was discovered to cause a similar threat in the new version of HTTP by creating a large payload at the back-end data link [12]. To get a higher detection rate for low-rate DDoS in real time monitoring traffic, certain server parameters like payload, hop count, latency and packet counts are considered [13]. In the case of Pulse DDoS [14], detection was made using TCP's congestion control window, which was plotted over a log scale and showed exponential variation in pulses in a real-time network.

The vulnerabilities of the protocol resulted in excessive use of central processing unit (CPU) and memory, reduced web throughput, and packet drops when a slow-rate DoS attack was launched [15]. An approach to detect and mitigate slow rate DoS attacks called Slowloris over an SDN network on HTTP/2 webserver was proposed by using an approach called Slowloris detection and mitigation mechanism (SDMM) using the expectation of burst size of incoming traffic flow. This approach gave 98% accuracy in DoS detection for typical and slow networks [16]. This new version of HTTP requires explicit TLS support for better performance when encrypting and securing data [17]. The Imperva Defense Center's investigation revealed that server administrators could not switch to the new version of the protocol without adding an additional layer of security because the protocol is more vulnerable to attacks [18]. According to an investigation conducted by Cisco, working on the HTTP/2 protocol over QUIC will help in easy and faster browsing compared to that incorporated over TCP. The added feature of QUIC will support encryption and handshakes in a connection [19].

The impact of HTTP/2 on Web services is that the default encryption feature of HTTP/2 results in traffic hiding that affects many services, such as web caching and traffic classification [20]. Ling *et al.* [21] claims that multiplexing and flow control features of HTTP/2 are one of the major reasons for application layer DoS attacks over HTTP/2, termed "H2DoS." He states that the severity of these types of attacks is high on web servers, thereby consuming resources over the entire network.

According to research work in [22], HTTP/2 helps to save energy consumption for mobile devices, thereby improving the performance of browsing. With the introduction of HTTP/2, studies proved that it is slightly better than the SPDY protocol at adapting to cellular networks but also has a negative impact on packet loss when adapted to mobile technology. A forensic report on security breaches over web surfing proved that HTTP/2 is more useful in the cellular field as its users can adapt to it more easily with minimal effort. A security analysis on signaling over 5G networks conducted by Hu *et al.* in [23] explains that service-based architecture in 5G network architecture is a major risk area for signaling. He discovered that the service-based architecture (SBA), when adapted for the HTTP/2 protocol, provides more security than other versions of HTTP by conducting testing over the 5G network using techniques such as fuzzing and model-based approach.

According to Sikora *et al.* in [24], precise detection of denial of service is a critical means of protecting cyber networks from attacks. In this work, an experimental generator was used to generate DoS on an HTTP/2-based Apache 2 server, which proved that, except for slow preface attacks, other sorts of DoS attacks could easily be traced over HTTP/2. This helps in securing networks from cyber threats. With precise detection, normal DoS could be recognized in a network, but according to [25], there could be a false alarm indicating that the network is safe in case of a stealthy denial of service attack. Hence, he proposes using ML techniques such as naive Bayes (NB), decision trees (DT), and support vector machines (SVMs) to detect stealthy traffic on HTTP/2 web servers. The study in [26], [27] talk about SDN performance analysis and traffic flow management. The background work on Slow DoS attacks, HTTP/1.1 and HTTP/2 protocols presented in this section has been a motivation for the proposed methodology of this paper.

2. PROPOSED METHOD

The proposed method is intended to detect and alleviate Slowloris attack on traditional networks and SDN having servers enabled with HTTP/1.1 and HTTP/2. The methodology is based on analyzing server parameters such as time for the server to connect to a webpage, latency in getting the response from the clients, page load time, the response-reply gap between the server and clients and inter-packet arrival time at the server. A threshold is set using Monte Carlo model to detect malicious activity based on server connect to a web page and latency. Server parameters are collected using the Apache benchmark (ab) tool [28] and h2load [29].

2.1. Experiment set up for traditional network

The architecture of the system comprises virtual machines installed on a host machine. It consists of multiple clients and a single server connected through a TCP connection via a Wi-Fi network having a 72 Mbps link speed. The server virtual machine (VM) is installed with Apache2 with HTTP/2 enabled. The clients and server VMs are connected through TCP connection having a secured socket layer (SSL) certificate, thus ensuring transport layer security (TLS) support over TCP. Among the multiple clients, one is made as a malicious client. This malicious client along with the genuine clients, processes their requests to the server over TLS enabled TCP. On analyzing server parameters and the result is found to be greater than the defined threshold, that particular IP address of the attacker client is isolated from the communication network as shown in Figure 3.

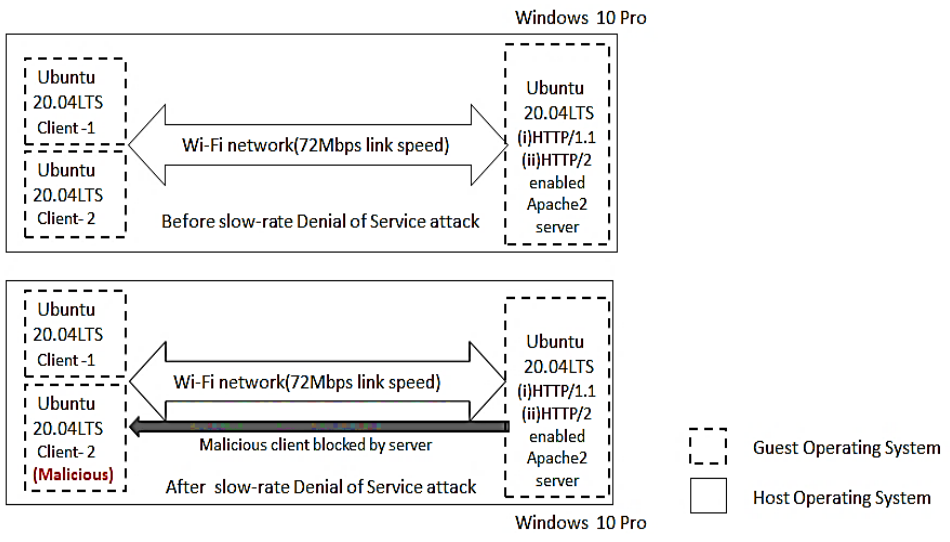

Windows 10 Pro
Ubuntu 20.04LTS Client-1
Ubuntu 20.04LTS Client-2
Wi-Fi network(72Mbps link speed)
Before slow-rate Denial of Service attack
Ubuntu 20.04LTS (i)HTTP/1.1 (ii)HTTP/2 enabled Apache2 server
Ubuntu 20.04LTS Client-1
Ubuntu 20.04LTS Client-2 (Malicious)
Wi-Fi network(72Mbps link speed)
Malicious client blocked by server
After slow-rate Denial of Service attack
Ubuntu 20.04LTS (i)HTTP/1.1 (ii)HTTP/2 enabled Apache2 server
Windows 10 Pro
Guest Operating System
Host Operating System

Figure 3. Architectural overview of the designed prototype with HTTP/1.1 (or) HTTP/2 enabled server test cases

2.2. Experiment set up SDN

An SDN is set up in a lab environment using Ryu controller, Zodiac FX OpenFlow switch [30], Apache2 servers, legitimate clients and attackers. Apache2 servers enabled with HTTP/1.1 and HTTP/2 are used in specific. The topology used in this work is mentioned in Figure 4.

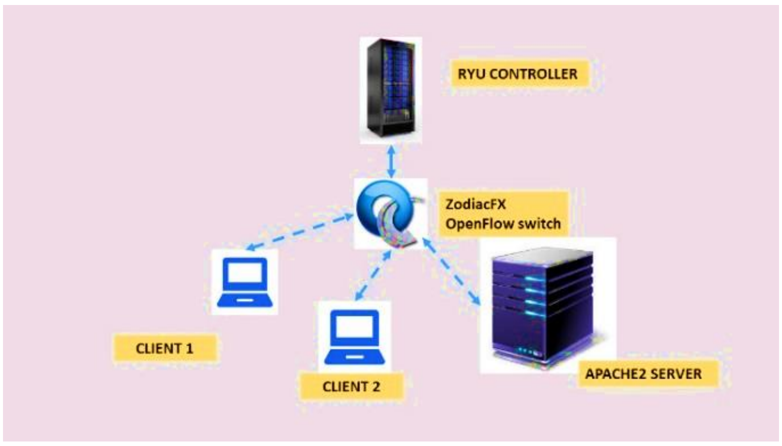


Figure 4. SDN topology

2.3. HTTP/2 server parameter analysis tools

Apache benchmark tool is a load testing and benchmarking tool. This tool is a simple command-line computer program for HTTP or HTTPS web servers. This tool helps us to know the number of requests processed per second by the web server.

H2load benchmark tool is benchmarking tool that can be used for both HTTP/2 and HTTP/1.1 supported web calls. It also helps with verifying SSL/TLS certification. h2load uses non-blocking i/o for making concurrent calls to target the GET/POST endpoint. Figure 5 explains the process of finding server parameters through benchmark tools. Here, the Apache benchmark is used to obtain parameters such as server connection time to a webpage, request processing time by the server, page load time, and latency in getting a response from the client, which is calculated mathematically by using the formula referred to as (1). The h2load benchmark is used to obtain parameters like the response and reply gap between endpoints and the inter-packet arrival time at the server.

$$Latency = \frac{(2 * \textit{requests processing time by a server})}{\textit{server connection time to a webpage}} \tag{1}$$

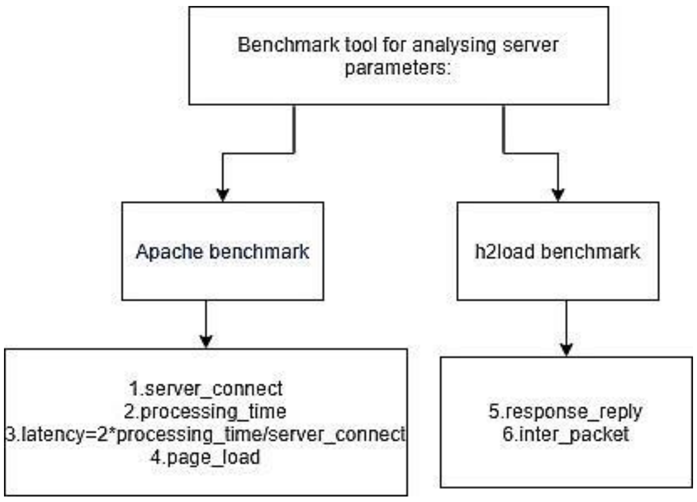


Figure 5. Benchmark tool used for server parameters

The workflow of the proposed method to analyze the performance of the Apache2 server using both the versions of HTTP, i.e., version 1.1 and 2, before and after malicious activity in both traditional and SDN, is shown in Figure 6. According to the flowchart, a server is connected to multiple clients in a network. Client-2 is intended to be an attacker and launches a Slowloris attack. The server script is executed on the server every 10 seconds, where parameters are obtained from the benchmark tool. With the obtained samples, a threshold is set for parameters such as time to connect to the webpage and latency in getting a response from the clients using the Monte Carlo model [31]. If the server detects a value greater than these two thresholds for their respective parameters or a timeout occurs, it is identified as malicious activity in the network. After the attacker source is identified, the server blocks the route of that client, thus breaking the connection between server and attacker.

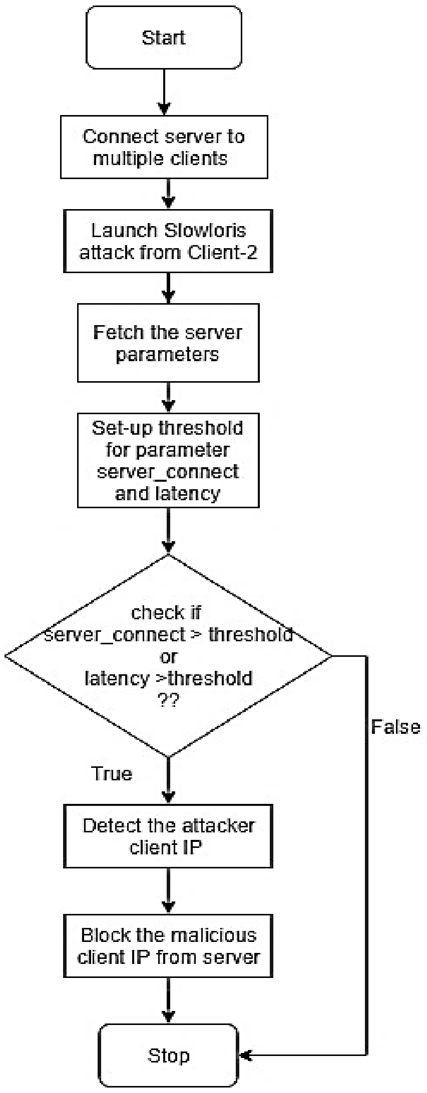

Start
Connect server to multiple clients
Launch Slowloris attack from Client-2
Fetch the server parameters
Set-up threshold for parameter server_connect and latency
check if server_connect > threshold or latency >threshold ??
False
True
Detect the attacker client IP
Block the malicious client IP from server
Stop

Figure 6. Workflow of the prototype

3. RESULTS AND DISCUSSION

This section shows the behavior of the network before and after launching the Slowloris attack on traditional networks and SDN on HTTP/1.1 and HTTP/2. The results are obtained using the topology shown in Figures 3 and 4. The changes in server parameters before and after Slowloris attack in traditional network and SDN are graphically depicted in Figures 7 to 12 and Tables 1 to 6. The graphs show that when an HTTP/1.1 server is attacked, it stops responding. It is indicated by 0. The HTTP/2 enabled server continues to respond instead of coming to a halt because multiple connections from the same client are multiplexed into the same stream.

3.1. The threshold values for attack detection

Attack detection is based on the identification of timeouts in the case of HTTP/1.1. On an HTTP/2 enabled server, an attack is identified when server connection time and latency for response exceed threshold values. With the obtained result of server parameters, the threshold is set for parameters, namely, server connection time to webpage and latency in getting a response from the server. The values of the threshold obtained using Monte Carlo simulation to detect the attack on HTTP/2 enabled server is indicated in Table 7.

3.2. Attack detection and termination from the network

When monitored regularly, if the value of the parameter defined turns out to be greater than the fixed threshold, then there is a detection of malicious behavior in the network. The detection and blocking of malicious clients from server HTTP/1.1 and HTTP/2 are as shown in Figures 13 and 14 respectively. Latency and server connection time are shown in Tables 8 and 9 respectively.

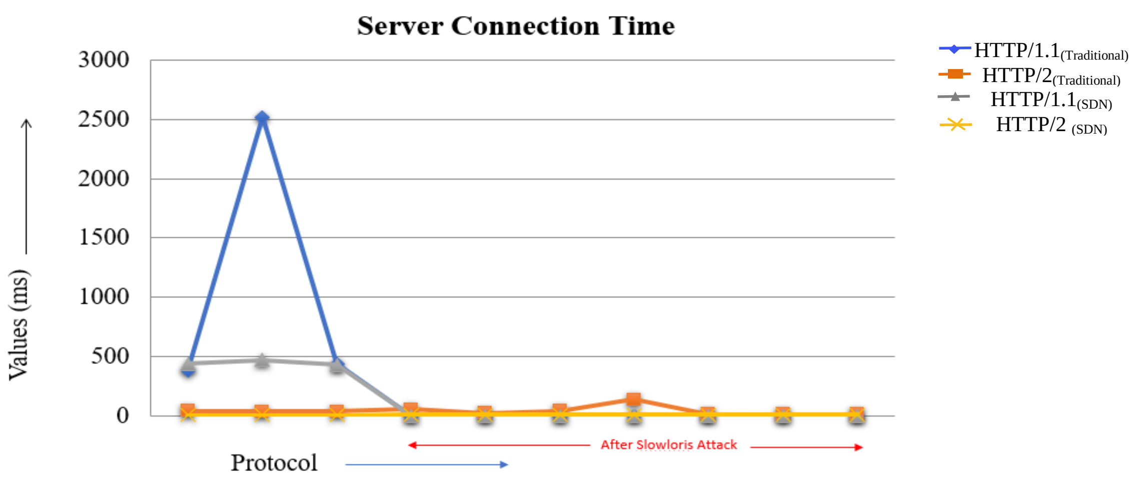


Figure 7. Server connections to webpage in protocol HTTP/1.1 and HTTP/2

Table 1. Values at different instances for server connection time

	Protocol	Server connection time (s) at intervals of 10 sec									
					← After Slowloris Attack →						
◆	HTTP/1.1(Traditional)	381	2520	437	0	0	0	0	0	0	0
■	HTTP/2(Traditional)	41	36	35	53	23	43	137	12	15	15
▲	HTTP/1.1(SDN)	439	468	430	0	0	0	0	0	0	0
✕	HTTP/2 (SDN)	4	5	4	11	10	9	9	8	10	10

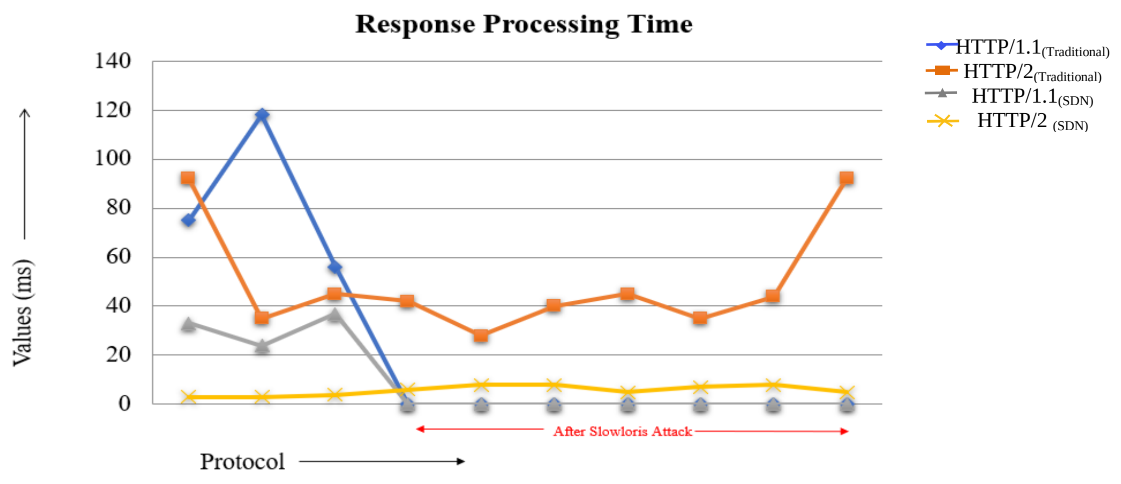


Figure 8. Response processing time in protocol HTTP/1.1 and HTTP/2

Table 2. Values at different instances for response processing time (s)

Protocol	Response processing time (s) at intervals of 10 sec									
			← After Slowloris Attack →							
◆ HTTP/1.1$_{(Traditional)}$	75	118	56	0	0	0	0	0	0	0
■ HTTP/2$_{(Traditional)}$	92	35	45	42	28	40	45	35	44	92
▲ HTTP/1.1$_{(SDN)}$	439	468	37	0	0	0	0	0	0	0
✕ HTTP/2 $_{(SDN)}$	4	5	4	6	8	8	5	7	8	5

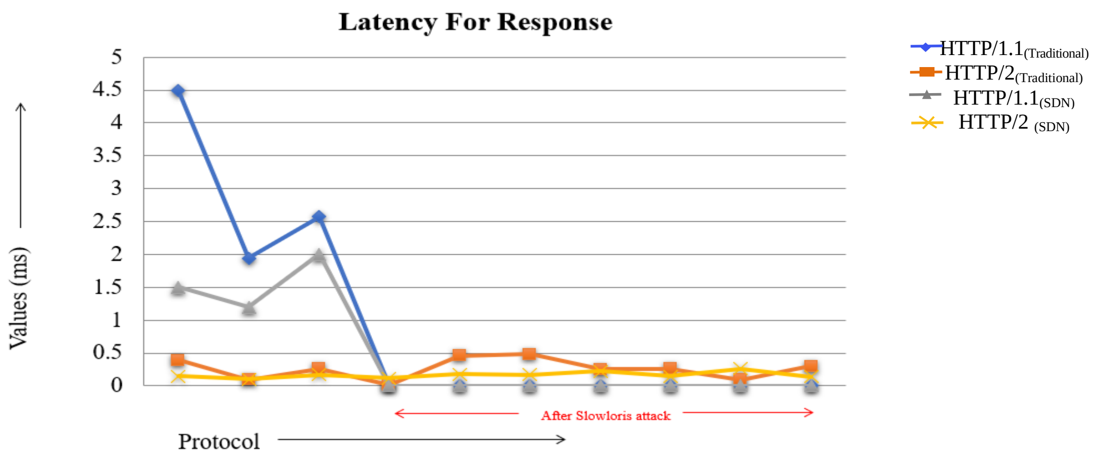


Figure 9. Latency for a response from the server in protocol HTTP/1.1 and HTTP/2

Table 3. Values at different instances for latency for response (s)

Protocol	Latency for response (s) at intervals of 10 sec									
			← After Slowloris Attack →							
◆ HTTP/1.1$_{(Traditional)}$	4.49	1.94	2.57	0	0	0	0	0	0	0
■ HTTP/2$_{(Traditional)}$	0.39	0.09	0.26	0.01	0.46	0.48	0.25	0.26	0.09	0.3
▲ HTTP/1.1$_{(SDN)}$	1.5	1.2	2	0	0	0	0	0	0	0
✕ HTTP/2 $_{(SDN)}$	0.15	0.1	0.17	0.12	0.18	0.17	0.22	0.15	0.26	0.14

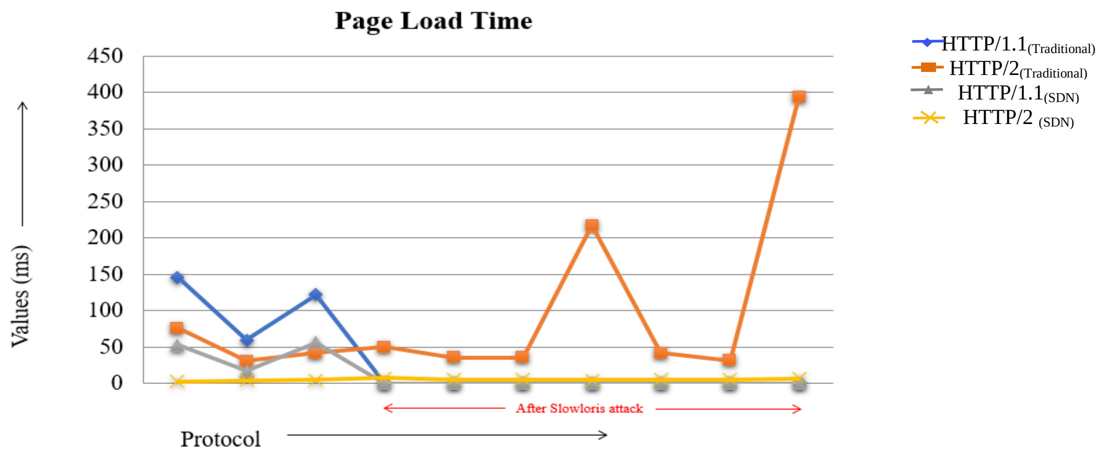


Figure 10. Page load time in protocol HTTP/1.1 and HTTP/2

Table 4. Values at different instances for page load time (s)

Protocol	Page load time (s) at intervals of 10 sec									
			← After Slowloris Attack →							
◆ HTTP/1.1$_{(Traditional)}$	146	60	122	0	0	0	0	0	0	0
■ HTTP/2$_{(Traditional)}$	76	31	42	50	36	36	217	42	32	394
▲ HTTP/1.1$_{(SDN)}$	53	17	56	0	0	0	0	0	0	0
✕ HTTP/2 $_{(SDN)}$	3	4	5	8	6	6	5	6	6	7

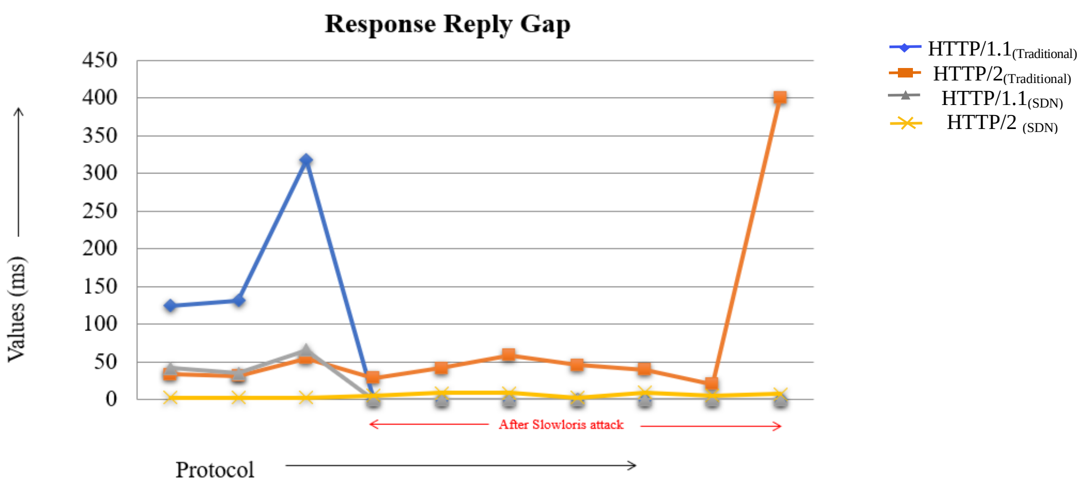


Figure 11. Response-reply gap between endpoints in protocol HTTP/1.1 and HTTP/2

Table 5. Values at different instances for response-reply gap (s)

	Protocol	Response-reply gap (s) at intervals of 10 sec									
					← After Slowloris Attack →						
◆	HTTP/1.1(Traditional)	125	132	318	0	0	0	0	0	0	0
■	HTTP/2(Traditional)	34	32	55	29	42	59	46	40	21	400
▲	HTTP/1.1(SDN)	42	35	66	0	0	0	0	0	0	0
✕	HTTP/2 (SDN)	3	3	3	6	9	9	3	10	6	8

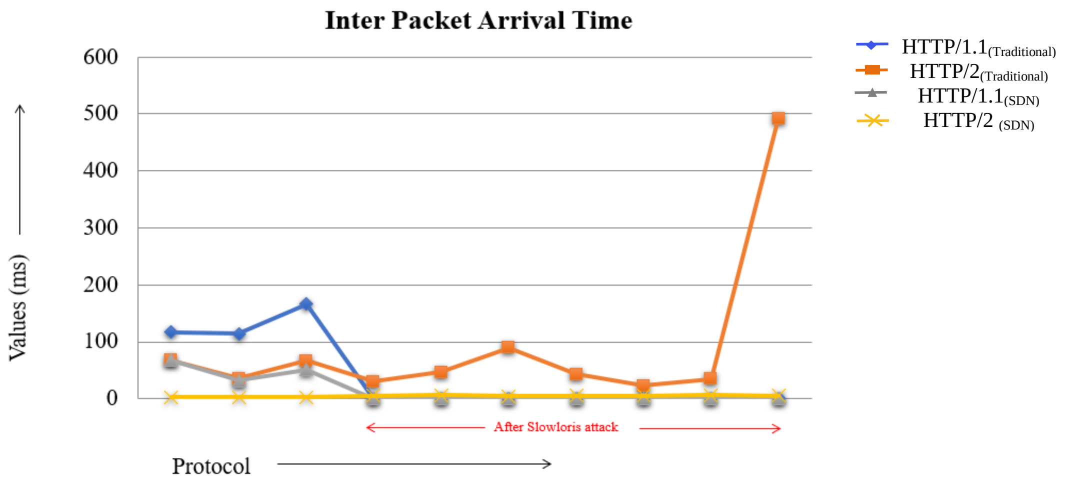


Figure 12. Inter-packet arrival time at server in protocol HTTP/1.1 and HTTP/2

Table 6. Values at different instances for inter packet arrival time (s)

	Protocol	Inter packet arrival time (s) at intervals of 10 sec									
					← After Slowloris Attack →						
◆	HTTP/1.1(Traditional)	117	132	318	0	0	0	0	0	0	0
■	HTTP/2(Traditional)	68	36	67	30	47	90	43	23	35	492
▲	HTTP/1.1(SDN)	68	32	51	0	0	0	0	0	0	0
✕	HTTP/2 (SDN)	3	3	3	6	7	5	6	6	7	6

Table 7. Threshold value for protocol HTTP/2

Server metrics	HTTP/2 Threshold (ms)
server connect to a webpage	130
latency for server response	0.4

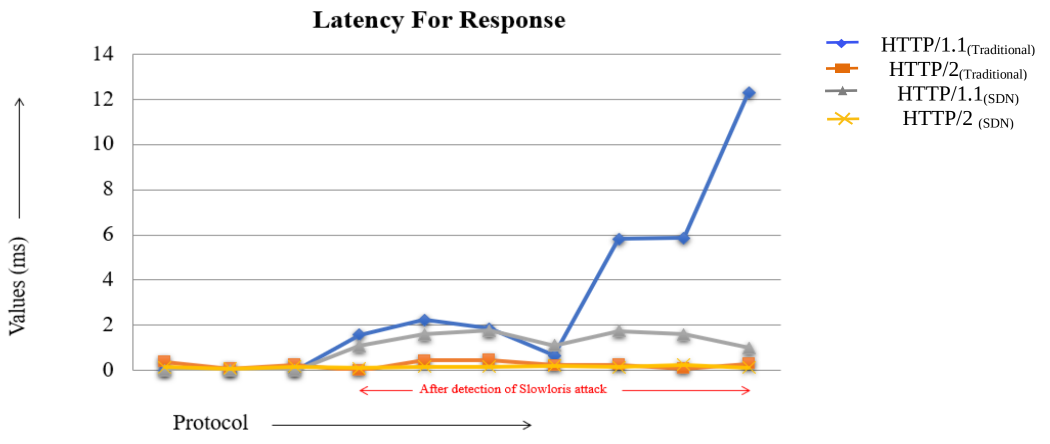


Figure 13. Slowloris attack detection and termination in HTTP/1.1 protocol

Table 8. Values at different instances for latency for response (s)

Protocol	Latency for response (s) at interval of 10 sec									
		← After detection of Slowloris attack →								
◆ HTTP/1.1$_{(Traditional)}$	0	0	0	1.58	2.24	1.86	0.66	5.83	5.87	12.3
■ HTTP/2$_{(Traditional)}$	0.39	0.09	0.26	0.01	0.46	0.48	0.25	0.26	0.09	0.3
▲ HTTP/1.1$_{(SDN)}$	0	0	0	1.09	16	1.77	1.11	1.75	1.6	1
✕ HTTP/2 $_{(SDN)}$	0.15	0.1	0.17	0.12	0.18	0.17	0.22	0.15	0.26	0.14

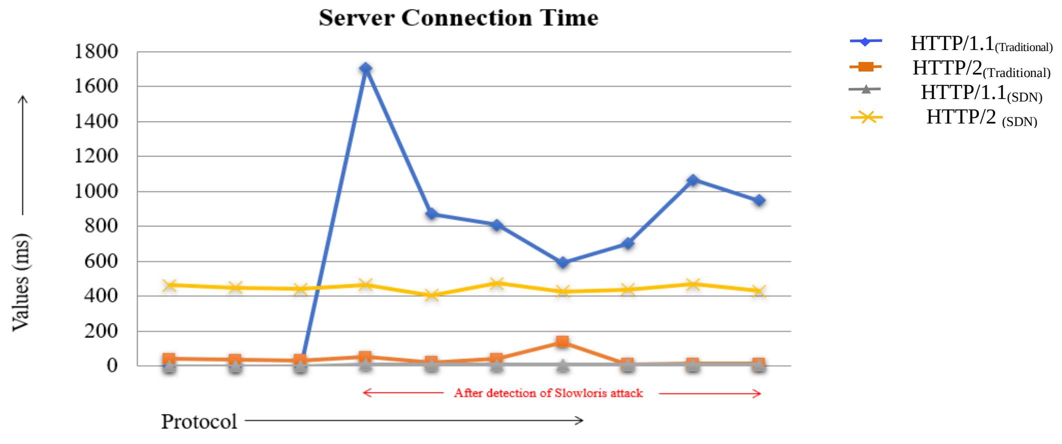


Figure 14. Slowloris attack detection and termination in HTTP/2 protocol

Table 9. Values at different instances for server connection time (s)

Protocol	Server connection time (s) at interval of 10 sec									
		← After detection of Slowloris attack →								
◆ HTTP/1.1$_{(Traditional)}$	0	0	0	1703	872	809	592	703	1067	949
■ HTTP/2$_{(Traditional)}$	41	36	35	53	23	43	137	12	15	15
▲ HTTP/1.1$_{(SDN)}$	0	0	0	11	10	9	9	8	10	10
✕ HTTP/2 $_{(SDN)}$	463	449	442	466	406	475	427	439	468	430

The graphs indicate the performance of the server after attack detection. After the malicious client is disconnected from the server, it starts responding normally. From the above experimental results, it is observed that the performance of the HTTP/2 protocol is way better than that of HTTP/1.1. With the introduced slow-rate DoS attack on the network, the HTTP/2 protocol has effectively proved that with multiple connections opened at the server, the protocol is still capable of high-speed data transfer as the per client's requests. SDN and traditional network respond in a similar manner to the Slowloris attack for HTTP/1.1 and HTTP/2 protocols. However, SDN's centralized architecture offers many advantages with respect to security of the entire network, which makes enterprise threat detection and mitigation easier.

3.3. Security benefits of SDN architecture

Today's enterprise networks face the challenge of handling evolving threats. Advanced enterprise and architecture and network security models are required to meet this challenge. At this point of advancement, SDN is a better choice to respond to these challenges. SDN offers a holistic view of the entire network from the controller. With the data and control planes separated, it provides greater flexibility to program the network at the controller.

The SDN architecture allows detection of malicious traffic on a granular basis rather than scanning the entire network. Network engineers can create special categories of devices that require different levels of security. Security updates can be easily deployed to these devices by programming the controller. The controller takes the responsibility of applying the updated security policies to the entire network without the intervention of the network engineer. Although SDN provides these security advantages, it is at the risk of a single point attack at the controller. Therefore, strong security policies and rules have to be enforced to prevent unauthorized access to the controller.

4. CONCLUSION

This paper investigates the performance of HTTP/1.1 and HTTP/2 enabled servers when subjected to a Slowloris attack on SDN and traditional networks. The performance of the HTTP/2 protocol was comparatively better than that of HTTP/1.1 concerning time parameters such as latency in getting a response from clients, page load time, response-reply gap, and inter-packet arrival time. Using the obtained values of these parameters, a threshold was set for the HTTP/2 server using a Monte Carlo model for attack detection. An attack was detected on the HTTP/1.1 server based on timeouts. The identified attacker was blocked from the network. The advantages offered by SDN architecture with respect to security were highlighted in this paper.

REFERENCES

[1] R. Deb and S. Roy, "A comprehensive survey of vulnerability and information security in SDN," *Computer Networks*, vol. 206, Apr. 2022, doi: 10.1016/j.comnet.2022.108802.

[2] S. Winkel and C. Walker, "Network forensics and HTTP/2." SANS Institute: InfoSec Reading Room, 2015.

[3] H. de Saxce, I. Oprescu, and Y. Chen, "Is HTTP/2 really faster than HTTP/1.1?," in *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Apr. 2015, pp. 293–299, doi: 10.1109/INFCOMW.2015.7179400.

[4] A. Praseed and P. S. Thilagam, "Fuzzy request set modelling for detecting multiplexed symmetric DDoS ttacks on HTTP/2 servers," *Expert Systems with Applications*, vol. 186, Dec. 2021, doi: 10.1016/j.eswa.2021.115697.

[5] E. Adi, Z. A. Baig, P. Hingston, and C.-P. Lam, "Distributed denial-of-service attacks against HTTP/2 services," *Cluster Computing*, vol. 19, no. 1, pp. 79–86, Mar. 2016, doi: 10.1007/s10586-015-0528-7.

[6] L. M. Bach, B. Mihaljevic, and A. Radovan, "Exploring HTTP/2 advantages and performance analysis using Java 9," in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, May 2017, pp. 1522–1527, doi: 10.23919/MIPRO.2017.7973663.

[7] E. Adi, Z. Baig, C. P. Lam, and P. Hingston, "Low-rate denial-of-service attacks against HTTP/2 services," in *2015 5th International Conference on IT Convergence and Security (ICITCS)*, Aug. 2015, pp. 1–5, doi: 10.1109/ICITCS.2015.7292994.

[8] I. Grigorik, "Transport layer security (TLS)," *High Performance Browser Networking; O'Reilly Media: Sebastopol, CA, USA*, 2013.

[9] M. Belshe and R. Peon, "Hypertext transfer protocol version 2 (HTTP/2)," RFC 7540, May 2015, doi: 10.17487/RFC7540.

[10] E. Adi, "Denial-of-service attack modelling and detection for HTTP/2 services," Master Thesis, Edith Cowan University, 2017.

[11] N. Tripathi and N. Hubballi, "Slow rate denial of service attacks against HTTP/2 and detection," *Computers & Security*, vol. 72, pp. 255–272, Jan. 2018, doi: 10.1016/j.cose.2017.09.009.

[12] D. Beckett and S. Sezer, "HTTP/2 Tsunami: Investigating HTTP/2 proxy amplification DDoS attacks," in *2017 Seventh International Conference on Emerging Security Technologies (EST)*, Sep. 2017, pp. 128–133, doi: 10.1109/EST.2017.8090411.

[13] M. Baskar, J. Ramkumar, C. Karthikeyan, V. Anbarasu, A. Balaji, and T. S. Arulananth, "Low rate DDoS mitigation using real-time multi threshold traffic monitoring system," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–9, 2021.

[14] G. Kaur, V. Saxena, and J. P. Gupta, "Detection of TCP targeted high bandwidth attacks using self-similarity," *Journal of King Saud University - Computer and Information Sciences*, vol. 32, no. 1, pp. 35–49, Jan. 2020, doi: 10.1016/j.jksuci.2017.05.004.

[15] Y. Zhang and Y. Shi, "A slow rate denial-of-service attack against HTTP/2," in *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*, Dec. 2018, pp. 1388–1391, doi: 10.1109/CompComm.2018.8780763.

[16] P. M. John and R. M. B. K. Nagappasetty, "An approach for slow distributed denial of service attack detection and alleviation in software defined networks," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 25, no. 1, pp. 404-413, Jan. 2022, doi: 10.11591/ijeecs.v25.i1.pp404-413.

[17] M. Suresh, P. P. Amritha, A. K. Mohan, and V. A. Kumar, "An investigation on HTTP/2 security," *Journal of Cyber Security and Mobility*, vol. 7, no. 1, pp. 161–189, 2018, doi: 10.13052/jcsm2245-1439.7112.

[18] S. Dulce and A. Shulman, "Man in the Cloud (MITC) attacks." https://www.slideshare.net/Imperva/maninthecloudattacksfinal (accessed Jan. 20, 2022).

[19] C. Pearce and C. Vincent, "HTTP/2 & QUIC-teaching good protocols to do bad things," *Blackhat, USA*, 2016.

[20] N. Ramadan and I. Mohamed, "Impact of implementing HTTP/2 in web services," *International Journal of Computer Applications*, vol. 147, no. 9, pp. 27–32, Aug. 2016, doi: 10.5120/ijca2016911182.

[21] X. Ling, C. Wu, S. Ji, and M. Han, "H DoS: An application-layer DoS attack towards HTTP/2 protocol," *Security and Privacy in Communication Networks: 13th International Conference, SecureComm 2017,* 2018, pp. 550–570.

[22] M. A. Abdillahi, U. Dossetov, and A. Saqib, “Performance evaluation of http/2 in modern web and mobile devices,” *American Journal of Engineering Research*, vol. 6, no. 4, pp. 40–45, 2017.
[23] X. Hu, C. Liu, S. Liu, W. You, and Y. Zhao, “Signalling security analysis: Is HTTP/2 secure in 5G core network?,” in *2018 10th International Conference on Wireless Communications and Signal Processing (WCSP)*, Oct. 2018, pp. 1–6, doi: 10.1109/WCSP.2018.8555612.
[24] M. Sikora, R. Fujdiak, K. Kuchar, E. Holasova, and J. Misurec, “Generator of slow denial-of-service cyber attacks,” *Sensors*, vol. 21, no. 16, p. 5473, Aug. 2021, doi: 10.3390/s21165473.
[25] E. Adi, Z. Baig, and P. Hingston, “Stealthy denial of service (DoS) attack modelling and detection for HTTP/2 services,” *Journal of Network and Computer Applications*, vol. 91, pp. 1–13, Aug. 2017, doi: 10.1016/j.jnca.2017.04.015.
[26] T. A. Assegie, “Performance analysis of emulated software defined wireless network,” *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 16, no. 1, pp. 311–3118, Oct. 2019, doi: 10.11591/ijeecs.v16.i1.pp311-318.
[27] H. T. Zaw and A. Maw, “Traffic management with elephant flow detection in software defined networks (SDN),” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 4, pp. 3203-3211, Aug. 2019, doi: 10.11591/ijece.v9i4.pp3203-3211.
[28] Apache, “ab - Apache HTTP server benchmarking tool - Apache HTTP Server Version 2.4” Apache HTTP Server Project, 2009. https://httpd.apache.org/docs/2.4/programs/ab.html (accessed Feb. 28, 2022).
[29] T. Tsujikawa, “HTTP/2 C library and tools,” Github. [Online]. Available: https://github.com/nghttp2/nghttp2 (accessed Nov. 01, 2021).
[30] P. Zanna, “Zodiac FX – the world’s smallest, most affordable OpenFlow SDN switch is now on Kickstarter.” Northbound Networks. [Online]. Available: https://northboundnetworks.com/pages/zodiac-fx-the-world-s-smallest-most-affordable-openflow-sdn-switch-is-now-on-kickstarter (accessed Jan. 01, 2022).
[31] C. Mooney, *Monte Carlo simulation*. 2455 Teller Road, Thousand Oaks California 91320 United States of America: SAGE Publications, Inc., 1997.

BIOGRAPHIES OF AUTHORS

Anusha A. Murthy received the B.Eng. degree in Electronics and Communication Engineering in 2019 and the M. Tech degree in computer networking engineering in 2021 from Visveshwaraya Technological University, Belagavi. She has industry experience on automating test cases for regression in Kuberneters platform from Nokia Network India Private Limted. Currently, she is working on switching solutions using VLAN as Software Engineer in Capgemini Engineering, Bengaluru. Her research interests include protocol security, cyber security using ML technique, Kuberneters, Docker, VMWare AWS, Cloud computing, python-based programming related to network security, ethical hacking. She can be contacted at email: anushamurthy183@gmail.com.

Prathima Mabel John is Assistant Professor at Dayananda Sagar College of Engineering, Visvesvaraya Technoligical University (VTU), Bengaluru, Karnataka, India. She received her Bachelor of Engineering and Master of Technology degree in Computer Science and Engineering from VTU, Belagavi, Karnataka, India. She is currently pursuing Ph D from VTU, Belagavi, Karnataka, India. She has about 13 years of experience in teaching and industry together. Her areas of interest are computer networks, SDN, mobile networks, network security and machine learning. She can be contacted at email: prathimamabel-ise@dayanandasagar.edu.

Rama Mohan Babu Kasturi Nagappasetty is currently working as Professor in the Department of Information Science and Engineering at Dayananda Sagar College of Engineering, Bengaluru, India. He obtained his B. Tech in Computer Engineering from Mangalore University, India, M.S from BITS-PILANI, India and PhD from Dr. MGR University, India. His areas of interest are computer networks, wireless mobile networks, SDN and network security. He can be contacted at email: ramamohanbabu-ise@dayanandasagar.edu.