

An approach for slow distributed denial of service attack detection and alleviation in software defined networks

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 Jun 22, 2021

Revised Nov 6, 2021

Accepted Dec 1, 2021

Keywords:

Denial of service

Expectation of burst size

HTTP2

Slowloris

Software defined network

ABSTRACT

Over the last few years, the need for programmable networks has captured the interest of industrialists and academicians. It has led to the development of a paradigm called software defined network (SDN). It separates the network intelligence into the control plane and forwarding logic into the data plane. This architecture gives scope to various security issues of which denial of service (DoS) is the most common and challenging to detect. This paper focuses on the detection and mitigation of a slow DoS attack called Slowloris on Apache2 server in SDN based networks. The proposed solution is called Slowloris detection and mitigation mechanism (SDMM). Mininet, an emulator, and SimpleHTTPServer are used for simulation and the same is implemented using Zodiac FX OpenFlow switch, Ryu controller and Apache2 server. SDMM algorithm detects and mitigates prolonged Slowloris attack in typical networks as well as in slow networks with low bandwidth and high delay in 240-280s with an accuracy of 100% and 98% respectively. It uses expectation of burst size as a key factor for detection.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Prathima Mabel John

Department of Information Science and Engineering, Dayananda Sagar College of Engineering

Visvesvaraya Technological University

Bengaluru, Karnataka 560091, India

Email: prathimamabel-ise@dayanandasagar.edu

1. INTRODUCTION

Software defined network (SDN) is a new networking paradigm that focuses on separating the control plane from the data plane [1]. The control plane and data plane communicate with each other using OpenFlow protocol. Controller is a device in the control plane which houses the intelligence of the entire network. This centralized architecture poses a greater vulnerability to attacks. If the controller is compromised, the entire network can be manipulated by attackers. Therefore, security is one of the major concerns for SDN. This paper is focused on improving security at the controller. This work focuses on one of the most popular attacks called the denial-of-service attack. A popular category of denial of service (DoS) attack is slow DoS attack which is characterized by sending a small amount of data over successfully established connections for a prolonged time period. A large number of such connections makes the victim reach its saturation state. This makes the server unavailable for legitimate clients resulting in DoS. These attacks are hard to detect as the attacker sends legitimate traffic and not corrupted data over legitimate connections.

Slow DoS attacks are application-level protocol attacks. Slowloris and slow hyper text transfer protocol (HTTP) POST are the common Slow HTTP DoS attacks tools. This paper presents a Slow HTTP DoS attack defence mechanism that detects and mitigates Slowloris attack. It works by allowing a single

machine to bring down another machine's server. It is initiated when the attacker establishes a lot of connections to the server and holds these connections for a long time by sending incomplete HTTP GET requests at regular intervals. The proposed solution is based on the calculation of expectation of burst size of incoming traffic. It is emulated using Mininet and SimpleHTTPServer. The solution is also implemented on a real network set up using Zodiac FX OpenFlow switch, Ryu controller and Apache2 web server.

The proposed solution is tested for the Slowloris attack on HTTP2 enabled Apache2 server. HTTP2 works by multiplexing concurrent requests from a client into a single connection. Slowloris in its default configuration does not bring the server down. However, Slowloris detection and mitigation mechanism (SDMM) is successful in detecting the attacker who opens the single connection with multiplexed concurrent requests. SDMM is successful in detecting attacks on HTTP2 enabled Apache2 server. The solution is tested with ZodiacFX switch, Ryu controller and Apache2 server enabled with HTTP2.

2. LITERATURE SURVEY

This section of the paper gives a clear picture of the work done in the area of distributed denial of service (DDoS) in SDN. Issues identified by other researchers along with the proposed solutions are mentioned here, which forms a basis for the proposed work in this paper. A comprehensive survey of evolution of SDN and research avenues is presented in [1]-[4]. These papers also bring forth various issues faced by SDN like scalability, performance, robustness, dependability, and security. Security is one such issue that has a vast scope for research and gained a lot of importance over years.

Praseed and Thilagam [5] present the points of vulnerability to attacks in SDN. They categorize attack scenarios into three categories: control plane specific, control channel specific and data plane specific. FRESCO by [6] is a framework for providing security services as modules. Table overflow attack is another threat to OpenFlow switches in the data plane. This attack is monitored and mitigated by [7] offers an extensive study of DoS attacks and their countermeasures in SDN. AVANT-GUARD proposed by [8] is another mechanism designed to counter DoS attacks caused by synchronize (SYN) flooding. FloodGuard proposed by [9] is a framework to prevent data-to-control plane saturation attack using proactive flow rule analyser and packet migration. [10] gives an extensive view of different types of DDoS attacks. DoS attacks are bifurcated as volumetric attacks, application layer attacks and slow DoS attacks. Some of the volumetric and application layer attacks are transmission control protocol (TCP) syn flooding, ping to death, user datagram protocol (UDP) flood, internet control message protocol (ICMP) flood, HTTP flood, and simple message transfer protocol (SMTP) flood. Slow DoS attacks utilize fewer resources, unlike flooding attacks. Some HTTP based slow attacks are slow HTTP header (Slowloris), slow HTTP POST attack (RUDY), and slow read attack. Slow HTTP header and Slow HTTP POST are generated by tools called Slowloris and RUDY respectively.

Methods to counter slow attacks are proposed by [10]. Slow HTTP header attacks like Slowloris, Slowloris-ng and Slow HTTP POST attacks are analyzed and detected based on long connections, low packet rate, packet distance uniformity, a combination of low packet rate and packet distance uniformity, low mean packet rate and low packet rate variance. Slow ternary content addressable memory (TCAM) exhaustion in SDN is explained by [11]. Slow attacks initiated by spoofed IPs fill up the OpenFlow tables of OpenFlow switches. A counter mechanism called selective defence for TCAM (SIFT) is proposed to decide whether a new rule should be installed or not. HTTP based slow attacks like Slowloris and SlowHTTPTest are analysed by [12]. Slow attacks on the Apache2 server are detected based on a threshold on the number of open connections. If threshold exceeds Slow HTTP DDoS attack defence method [SHDA] is triggered at the controller for the suspected clients.

A multi threshold monitoring system is proposed by [13] to improve low rate threat detection and mitigation. It computes low rate threat measure by considering payload, hop count, latency and packet count to detect low rate attacks [14]. analyses different types of pulsating attack. A detection method using self-similarity of incoming network traffic and wavelet based multi-resolution analysis is used to detect various types of pulsating attacks. HTTP2 based servers differ in the way in which simultaneous requests from a single client are handled [15]. presents a study of performance of HTTP1.1 and HTTP2 servers under DDoS attack. It was identified that HTTP2 are vulnerable to slow read attacks and HTTP flooding attacks.

Tripathi and Hubballi [16] proposes a method where attacks are detected using Chi-square statistic where statistics of traffic were collected over a period ΔT . Chi-square value is calculated over this traffic profile to test the hypotheses of whether the collected traffic belongs to normal or malicious traffic. The research propose solutions to detect and mitigate DDoS attacks in SDN based networks [17]-[22]. Explains how a non OpenFlow home gateway can be converted into a OpenFlow enabled network. The difference between traditional network and SDN and detection of elephant flows is brought about clearly in [23]. The research perform performance evaluation of the of emulating SDN and the impact of firewalls on throughput of the network [24], [25].

This background study forms a strong basis for studying the nature of attack and to design identification and mitigation technique proposed in SDMM. The efficiency of HTTP2 over HTTP1.1 presented by researchers has been a motivation to study the behavior of attack on HTTP2 enabled servers.

3. PROPOSED METHOD

The solution proposed in this paper is focused on detecting and mitigating slow DDoS attacks which are prolonged for a long time of more than 60s. Slowloris is one such attack in the Slow HTTP header attack category which is common and difficult to detect. In this work, a mechanism to detect and mitigate Slowloris attack on both HTTP1.1 and HTTP2 enabled Apache2 server called SDMM is proposed. This attack is intended to make the victim server unavailable for legitimate requests resulting in DoS. This is achieved when the attacker opens a lot of simultaneous legitimate connections after a successful TCP handshake with the victim server. Slowloris opens 150 connections to the Apache2 server by default. This value is not changed in this work. For the attack to be successful, these connections must be kept open for a long time. So, the attacker sends incomplete HTTP GET messages on these open connections every 15s if the attack is alive. The victim server becomes unavailable for legitimate clients as its thread pool is exhausted in this process.

The work proposed in this paper considers Slow DoS attack in HTTP2 apart from HTTP1.1. HTTP1.1 differs from HTTP2 in the way concurrent requests and responses are processed. HTTP1.1 processes requests and responses as plain text as opposed to HTTP2 which adds a binary framing layer that encapsulates all messages in binary format. Multiple requests from a single client are multiplexed into streams and sent on a single TCP connection. An HTTP2 enabled Apache2 server is used in this work for demonstrating the behaviour of Slowloris in HTTP2. Since all the 150 concurrent requests were multiplexed into a single TCP connection, the server was not brought down. However, the attacker is identified using SDMM based on the burst characteristics. SDMM is implemented in Ryu controller and has 2 phases: monitoring phase, detection and mitigation phase. The monitoring phase keeps track of the flow statistics of the Zodiac FX switch every 20s. Further detection and mitigation phase detects attack bursts from the collected flow statistics. Detection is based on the expectation of burst size. Once the attack is detected, the mitigation phase mitigates and blocks traffic coming in from the attacker. The attacker's mac address is added to a blacklist. Table 1 shows the assumptions and terminologies used in the algorithm. Overall functionality of SDMM is shown in the algorithm.

Table 1. Assumptions and thresholds

Term	Meaning /Threshold
C _i	i th Client
N _i	i th attack attempt
N _{th}	Attack attempt threshold = 3
No. of spikes per attack	4
No of concurrent connections by Slowloris	Simple HTTP Server = 7 Apache2 server = 150

Begin

Step 1: Monitor flow statistics of each of 4 consecutive flows every 20s.
 Step 2: Calculate burst size $b_i = b_{i+1} - b_i$, increment flow_counter (flow_counter tracks the no. of flows)
 Step 3: Calculate probability of burst $P(b_i)$ using (1)
 Step 4: Calculate Expectation of burst size $E(B)$ using (2)
 Step 5: Find standard deviation σ using (3)
 Step 6: Increment spike count N_i
 Step 7: For next 4 consecutive burst calculate $P(b_i)$ using (1)
 Step 8: Calculate Expectation of burst size $E(S)$ using (2) find $k = |E(B) - E(S)| / \sigma$
 //Detection and mitigation
 Step 9: if $E(B) - k \sigma \leq E(S) \leq E(B) + k \sigma$, GoTo Step 10
 Else goto Step 7
 Step 10: if $N_i = 1$ goto 11 else 12
 Step 11: Divert all the flows from attacker to server and vice versa to the controller.
 Step 12: Increment N_i
 Step 13: if $N_i \geq N_{th}$ GoTo Step 14 else goto Step 15
 Step 14: Client C_i is an attacker. Delete all the flow entries in the switch for client C_i and drop all the incoming packets from C_i .
 Step 15: if flow_counter > 13 goto Step 16 else goto Step 7
 Step 16: Client C_i is not an attacker. Install flows in the switch to send traffic buffered at the controller to the destination.

Monitoring phase is implemented by using a monitor which collects flow statistics from the flow table every 20s. The burst size is calculated by the difference in the byte count of every 2 consecutive flows. SDMM is designed for prolonged attacks which are extended for more than 60s. Therefore 4 bursts are considered to evaluate expectation of burst size. Let b_1, b_2, b_3, b_4 be consecutive bursts. The monitor's responsibility is to calculate the expectation of burst size $E(B)$ given by (2). The probability of burst b_i required to find $E(B)$ is given by (1). The standard deviation σ , is calculated using $P(b_i)$ and $E(B)$ as shown in (3),

$$P(b_i) = b_i / \sum_{i=1}^n b_i \quad (1)$$

where,

$P(b_i)$ - probability of burst b_i

b_i - i^{th} burst

n - No. of flows

expectation/mean of burst size is given by $E(B)$ for n no. of flows in (2).

$$E(B) = \sum_{i=1}^n b_i * P(b_i) \quad (2)$$

Standard deviation σ is given by formula in (3).

$$\sigma = \sqrt{\sum_{i=1}^n (b_i - (E(B)))^2 * P(b_i)} \quad (3)$$

$E(B)$ and σ is calculated for the first four consecutive flows and sets N_i to 1. The bursty nature of Slowloris attack is exploited to detect it. The bursts generated by the attack are consistent in size. Detection and mitigation phase is responsible to analyse if the next consecutive incoming traffic is from an attacker. For the next 4 consecutive bursts, $E(S)$ is calculated using (2). If $E(S)$ lies in the range $E(B) - k\sigma \leq E(S) \leq E(B) + k\sigma$, then it may be due to an attack. Here k is the tolerance factor used to adjust the minimum and maximum range of expected burst size. The client C_i is suspected to be an attacker and N_i is incremented considering this as an attack attempt.

At this point SDMM installs flow rules in the switch to divert the traffic coming from the suspected client C_i and the victim server to the controller. Controller starts buffering the incoming traffic from C_i and victim node. If the next 4 incoming bursts have $E(S)$ lying in the range $E(B) - k\sigma \leq E(S) \leq E(B) + k\sigma$, then N_i is incremented. This process repeats until N_i reaches a threshold N_{th} . When N_i reaches N_{th} for consecutive bursts, client C_i is identified as an attacker. The controller drops the buffered data and installs flow rules to drop all packets from C_i . Suppose $E(S)$ for consecutive bursts do not lie in the range $E(B) - k\sigma \leq E(S) \leq E(B) + k\sigma$, then the controller installs flow rules in the switch to forward the buffered data to the destined server.

SDMM detects and mitigates Slowloris attack in approximately 260s. The proposed method also considers attackers and legitimate clients in slow networks. Legitimate clients with slow networks are less likely to be falsely detected as attackers as they do not generate constantly sized consecutive bursts of data for a long time. Whereas attackers with slow networks generate irregular bursts in the beginning and constant bursts after they get connected to the server. These bursts are detected by SDMM to mitigate the attack.

4. EXPERIMENT AND RESULTS

This section details the experimental setup used for emulation as well as real network set up. This experiment considers clients in networks with typical network characteristics where all hosts can communicate without delay or congestion. It also considers clients in slow networks which has delays due to low bandwidth and congestion. Bandwidth and delay are varied in the emulation as well as real network set up for performance evaluation.

Table 1 shows the thresholds of various parameters chosen for this experiment. The number of spikes per attack attempt is chosen to be 4 as this work focuses on the Slowloris attack which extends for more than 60s. N_{th} , the attack attempt threshold, is chosen to be 3 to increase the accuracy of attack detection and to keep the detection time less than the default timeout of Apache2 server that is 300s. Continuous monitoring is done up to N_{th} attempts to reduce the rate of false positive detections.

4.1. Emulation using Mininet

Figure 1 shows the experimental set up using Open vSwitch (OVS), clients, Ryu controller and SimpleHTTP server. It consists of 50 clients, 1 SimpleHTTP server, 2 Open vSwitch and 1 Ryu controller. The experiment was conducted for typical networks as well as slow networks. Among 50 clients up to 4 attackers were detected in different cases as mentioned in the results. When a client C_i begins Slowloris

attack on the SimpeHTTP server, it opens 7 simultaneous connections. Consistent burst of bytes was generated byte the attacking client. The behavior of server under attack without SDMM is shown in Figure 2. The consecutively generated constant bursts can be clearly seen in the graph.

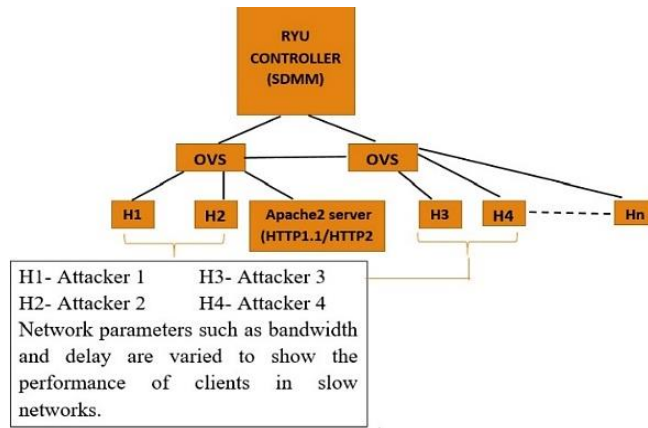


Figure 1. Mininet topology

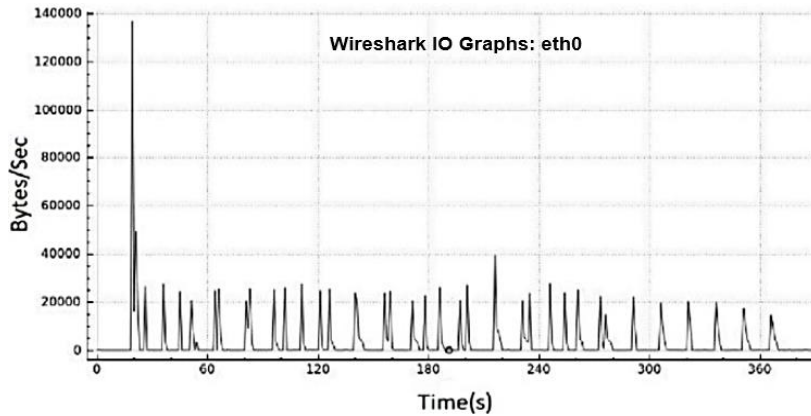


Figure 2. Server under Slowloris attack without SDMM

When SDMM was implemented the attack on the server was detected and mitigated successfully. Table 2 summarizes simulation results for typical network with 50 clients. It was successfully detected by SDMM within 260- 280s. The average burst of bytes that was generated in different cases are mentioned in the table. Figure 3 show the input output (IO) graph of the server when attack was detected using SDMM. It can be clearly seen that there is no packet transmission after 260s once the attack is detected.

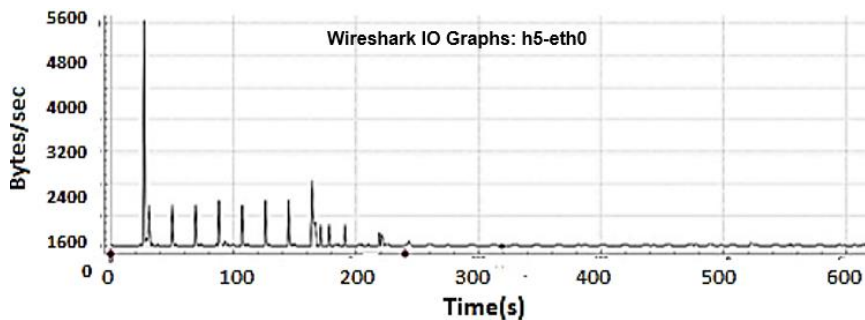


Figure 3. Slowloris attack detected on the server by SDMM in typical network scenario using Mininet

Table 2. Results of simulation using Mininet for typical network

Case	Attack scenario	Avg. Burst size	Detection time
1	Number of attackers=1	869 bytes	260s
2	Number of attackers=2	860 bytes	260s

In case 2, two attackers initiated Slowloris attack. Attacker 1 was able to occupy all the 7 seven connections simultaneously under ideal network conditions, leaving no threads for attacker 2. Hence, it was an unsuccessful attack attempt for attacker 2. Table 3 shows the results of emulation for clients in slow networks and single OVS. Slow network was emulated by varying bandwidth and delay of links between the client and OVS. Up to 4 attackers were detected. Each attacker was able to establish connection to the server using different number of sockets. Figure 4 shows server graph when attack is detected by SDMM. Packets are dropped at the client side after attack detection. Table 4 summarizes the results of emulation using 2 OVS and 50 clients in slow network. Figure 5 shows server behavior after attack detection. Packets are dropped at the client side after attack detection.

Table 3. Results of simulation using Mininet for clients in slow network with 1 switch

Case	Attack scenario	Avg. Burst size	Detection time
1	Number of attackers=1	934 bytes	260s
2	Number of attackers=2	298 bytes	260s
3	Number of attackers=4	320 bytes	260s

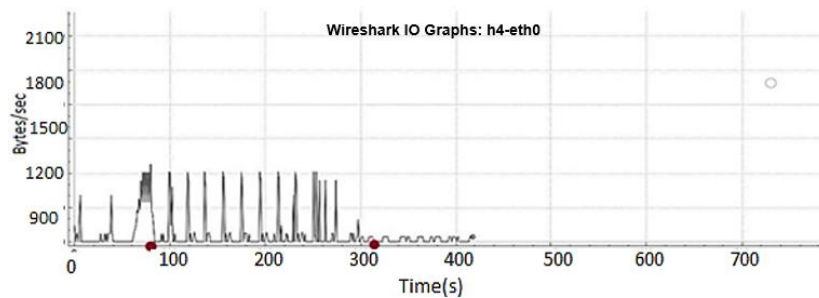


Figure 4. Slowloris attack detected on the server by SDMM in slow network scenario using Mininet

Table 4. Results of simulation using Mininet for clients in slow network using 2 switches

Case	Attack scenario	Avg. Burst size	Detection time
1	Number of attackers=1	530 bytes	260s
2	Number of attackers=4	290 bytes	280s

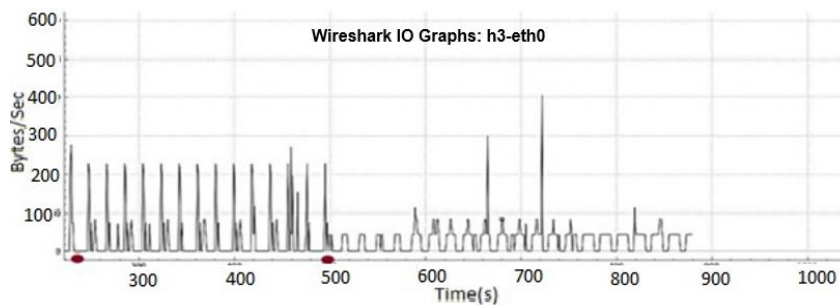


Figure 5. Slowloris attack detected on the server by SDMM in 2 switch slow network scenario using Mininet

4.2. Experiment in real network environment

Figure 6 shows the experimental set up using Zodiac FX switch, clients, Ryu controller and Apache2 server. It consists of 2 clients, 1 Apache2 server, one Zodiac FX switch and 1 Ryu controller. The experiment was conducted for typical networks as well as slow networks. Client Ci begins the Slowloris attack on the Apache2 server. It opens 150 simultaneous connections to the Apache2 server. The server’s thread pool gets

exhausted and becomes unavailable for other legitimate clients. Requests from other legitimate clients are timed out as the server is busy serving the attacker. Table 5 shows various scenarios considered for the experiment along with the average burst size generated by the attack. Figure 7 shows server behavior after detection.

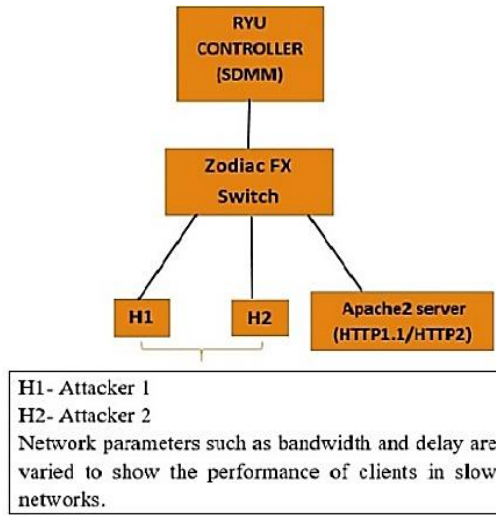


Figure 6. Topology of experiment using Zodiac FX switch

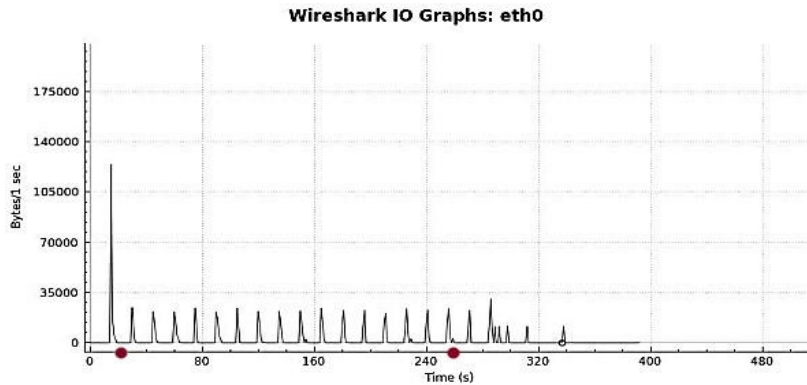


Figure 7. Slowloris attack detected on the server by SDMM in typical network scenario in real network

Table 5. Results of experiment using Zodiac FX in a typical network scenario with 2 clients

Case	Attack scenario	Avg. Burst size	Detection time
1	Number of attackers=1	23746 bytes	240s
2	Number of attackers=2	20617 bytes	260s

Table 6 shows the attack characteristics of 2 attackers in a slow network. In this scenario, the clients do not create 150 sockets at once due to the delay it experiences in the network. Instead, they create it at a slow pace till they acquire all the threads, attacking the server successfully. Figure 8 depicts attack detection on the server. In case 2, attacker 1 and attacker 2 connect to the server creating 129 and 127 sockets respectively. The server reaches its maximum connection limit of 256 (default Apache2 configuration). This makes the server unavailable for other clients.

Table 6. Results of experiment using Zodiac FX in a slow network scenario with 2 clients

Case	Attack scenario	Avg. Burst size	Detection Time
1	Number of attackers=1	27218 bytes	260s
2	Number of attackers=2	20285 bytes	260s

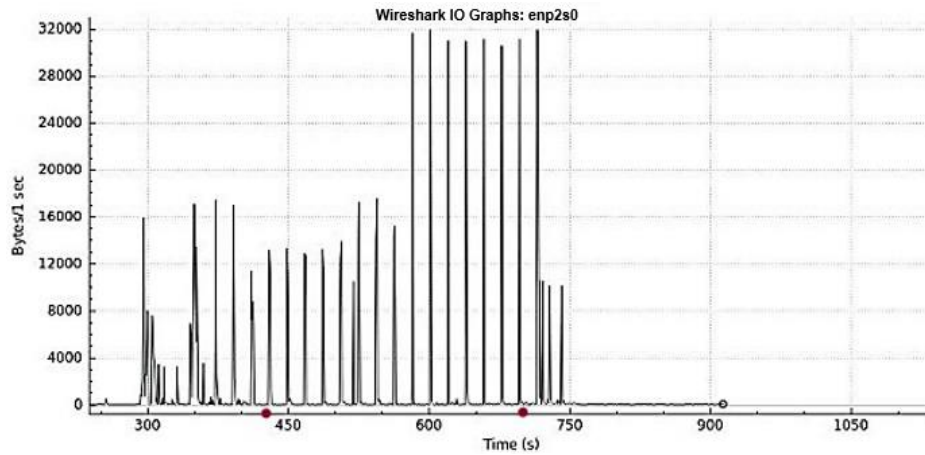


Figure 8. Slowloris attack detected on the server by SDMM in slow network scenario in real network

4.3. Experiment in real network environment for HTTP2 enabled server

Figure 9 shows the experimental set up using Zodiac FX switch, clients, Ryu controller and Apache2 server enabled with HTTP2. It consists of 2 clients, 1 Apache2 server, one Zodiac FX switch and 1 Ryu controller. Experiment was conducted for typical network. Results are as shown in Table 7 and Figure 9. The results clearly show that the attack is detected and mitigated in approximately 240-260s, earlier than the default timeout of the Apache2 server which is 300s. The attack behaviour of generating regular bursts of bytes is the key feature for attack detection in this work.

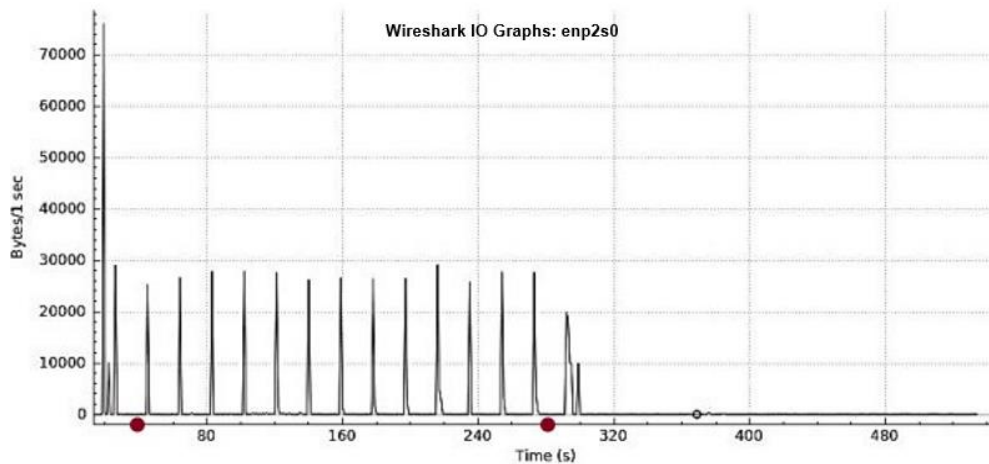


Figure 9. Slowloris attack detected on the HTTP2 enabled server by SDMM in typical network scenario

Table 7. Results of experiment for attack detection on HTTP2 enabled server and 2 clients using Zodiac FX switch

Case	Attack scenario	Avg. Burst size	Detection time
1	Number of attackers=1	26024 bytes	260s
2	Number of attackers=2	25260 bytes	260s

The results clearly indicate that the bursty behavior of the Slowloris can be used to identify and mitigate the attack. SDMM identifies the spikes and calculates $E(S)$, k and σ . It successfully detects and mitigates the attacker by identifying continuous spikes in the range $E(B)-k \sigma \leq E(S) \leq E(B)+k \sigma$. In order to measure the accuracy of the proposed system in detecting Slowloris attack, the balanced accuracy is calculated for a test set size of 20 for each scenario. Balance accuracy is given by (4) with true positive values (TP), true negative values (TN), false positive values (FP) and false negative values (FN).

$$\text{Balanced Accuracy} = \frac{TP}{TN+FN} + \frac{TN}{TN+FP} \quad (4)$$

Table 8 shows the balance accuracy of the test results obtained in this work. Compared to other proposed methods mentioned in section 2, accuracy is given preference over detection time in this proposed work where various network conditions are considered. Slow networks take time to generate constantly sized bursts because of varying bandwidth. Considering prolonged Slowloris attack in slow and ideal network scenarios, it is worthwhile to compromise over faster detection time to achieve greater accuracy. SDMM performed well with high rate of accuracy even in slow networks where bandwidth and delay keep fluctuating. In ideal networks, accuracy of SDMM was 100%. FP and FN values are very low. This proves that the proposed mechanism is successful enough to detect and mitigate attacks under various network conditions.

Table 8. Balanced accuracy of test scenarios

Test scenario	TP	FP	FN	TN	Balanced Accuracy
Emulation–Typical network	20	0	0	20	100%
Emulation– Slow network (Single switch)	18	0	2	20	95%
Emulation– Slow network (2 switches)	18	0	2	20	95%
Network setup with Zodiac FX –Typical network	20	0	0	20	100%
Network setup with Zodiac FX – Slow network	19	0	0	20	98%
Network setup with Zodiac FX –HTTP2 enabled server	19	0	0	20	98%

5. CONCLUSION

In this paper, slow DoS detection and mitigation (SDMM), a method to detect and mitigate Slowloris, a type of slow HTTP request attack, was proposed. Detection was based on the expectation of burst size of incoming flows. Flows were monitored by the monitoring phase to obtain the burst size of each incoming flow. The detection phase detected and mitigated the attack if 4 consecutive bursts were in the range of attack. The simulation was conducted in Mininet with Simple HTTP Server for clients in slow networks as well as typical networks. The same solution was implemented using the Zodiac FX switch, Ryu controller and Apache2 server. Attack was detected in almost all the scenarios in 260s on an average. The attack was also tested on HTTP2 enabled Apache2 server. Though the Slowloris attack in its default state could not bring the server down, the bursts initiated by the attacker were identified by SDMM. The client who initiated the attack was isolated by dropping any further traffic. Accuracy of the proposed system is evaluated by calculating balanced accuracy of test scenarios. It is found that SDMM detects attacks in typical network with an accuracy of 100% and in slow networks with an accuracy of 98%. There is a scope to extend this work for more clients and servers. It is limited to 2 clients and 1 server due to limitations in resource availability.





REFERENCES

- [1] A. L. V. Caraguay, L. I. B. Lopez, and L. J. G. Villalba, "Evolution and challenges of software defined networking," in *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, 2013: IEEE, pp. 1-7, doi: 10.1109/SDN4FNS.2013.6702542.
- [2] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: an intellectual history of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87-98, 2014, doi: 10.1145/2602204.2602219.
- [3] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network and openflow: From concept to implementation," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2181-2206, 2014, doi: 10.1109/COMST.2014.2326417.
- [4] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications surveys & tutorials*, vol. 16, no. 3, pp. 1617-1634, 2014, doi: 10.1109/SURV.2014.012214.00180.
- [5] C. Yoon *et al.*, "Flow wars: Systemizing the attack surface and defenses in software-defined networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3514-3530, 2017, doi: 10.1109/TNET.2017.2748159.
- [6] S. W. Shin, P. Porras, V. Yegneswara, M. Fong, G. Gu, and M. Tyson, "Fresco: Modular composable security services for software-defined networks," in *20th Annual Network & Distributed System Security Symposium*, 2013: Ndss.
- [7] T. Xu, D. Gao, P. Dong, C. H. Foh, and H. Zhang, "Mitigating the table-overflow attack in software-defined networking," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 1086-1097, 2017, doi: 10.1109/TNSM.2017.2758796.
- [8] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Avant-guard: Scalable and vigilant switch flow management in software-defined networks," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 413-424, doi: 10.1145/2508859.2516684.
- [9] H. Wang, L. Xu, and G. Gu, "Floodguard: A dos attack prevention extension in software-defined networks," in *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2015: IEEE, pp. 239-250, doi: 10.1109/DSN.2015.27.
- [10] R. Swami, M. Dave, and V. Ranga, "Software-defined networking-based ddos defense mechanisms," *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, pp. 1-36, 2019, doi: 10.1145/3301614.





- [11] T. A. Pascoal, Y. G. Dantas, I. E. Fonseca, and V. Nigam, "Slow TCAM exhaustion DDoS attack," in *IFIP International Conference on ICT Systems Security and Privacy Protection*, 2017: Springer, pp. 17-31, doi: 10.1007/978-3-319-58469-0_2.
- [12] T. Lukaseder, L. Maile, B. Erb, and F. Kargl, "Sdn-assisted network-based mitigation of slow ddos attacks," in *International Conference on Security and Privacy in Communication Systems*, 2018: Springer, pp. 102-121, doi: 10.1007/978-3-030-01704-0_6.
- [13] M. Baskar, J. Ramkumar, C. Karthikeyan, V. Anbarasu, A. Balaji, and T. Arulananth, "Low rate DDoS mitigation using real-time multi threshold traffic monitoring system," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1-9, 2021, doi: 10.1007/s12652-020-02744-y.
- [14] G. Kaur, V. Saxena, and J. 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, 2020, doi: 10.1016/j.jksuci.2017.05.004.
- [15] A. Praseed and P. S. Thilagam, "Multiplexed asymmetric attacks: Next-generation DDoS on HTTP/2 servers," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1790-1800, 2019, doi: 10.1109/TIFS.2019.2950121.
- [16] N. Tripathi and N. Hubballi, "Slow rate denial of service attacks against HTTP/2 and detection," *Computers & security*, vol. 72, pp. 255-272, 2018, doi: 10.1016/j.cose.2017.09.009.
- [17] N. M. A. Azim, S. F. Fahmy, M. A. Sobh, and A. M. B. Eldin, "A hybrid entropy-based DoS attacks detection system for software defined networks (SDN): A proposed trust mechanism," *Egyptian Informatics Journal*, vol. 22, no. 1, pp. 85-90, 2021, doi: 10.1016/j.eij.2020.04.005.
- [18] A. Sallam, A. Refaey, and A. Shami, "On the security of SDN: A completed secure and scalable framework using the software-defined perimeter," *IEEE Access*, vol. 7, pp. 146577-146587, 2019, doi: 10.1109/ACCESS.2019.2939780.
- [19] T. Wang, H. Chen, G. Cheng, and Y. Lu, "SDNManager: A safeguard architecture for SDN DoS attacks based on bandwidth prediction," *Security and Communication Networks*, vol. 2018, 2018, doi: 10.1155/2018/7545079.
- [20] S. Deng, X. Gao, Z. Lu, Z. Li, and X. Gao, "DoS vulnerabilities and mitigation strategies in software-defined networks," *Journal of Network and Computer Applications*, vol. 125, pp. 209-219, 2019, doi: 10.1016/j.jnca.2018.10.011.
- [21] J. G.-Brajones, J. C.-Murillo, J. F. V.-Valdés, and F. L.-Valero, "Detection and mitigation of dos and ddos attacks in iot-based stateful sdn: An experimental approach," *Sensors*, vol. 20, no. 3, p. 816, 2020, doi: 10.3390/s20030816.
- [22] S. Miano and F. Risso, "Transforming a traditional home gateway into a hardware-accelerated SDN switch," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 3, p. 2668, 2020, doi: 10.11591/ijece.v10i3.pp2668-2681.
- [23] 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, p. 3203, 2019, doi: 10.11591/ijece.v9i4.pp3203-3211.
- [24] S. Assegie and P. Nair, "Performance analysis of emulated software defined wireless network," *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 16, no. 1, pp. 311-318, 2019, doi: 10.11591/ijeecs.v16.i1.pp311-318.
- [25] H. Khairi, S. H. Ariffin, N. A. Latiff, K. M. Yusof, M. Hassan, and M. Rava, "The impact of firewall on TCP and UDP throughput in an openflow software defined network," *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 20, no. 1, pp. 256-263, 2020, doi: 10.11591/ijeecs.v20.i1.pp256-263.

BIOGRAPHIES OF AUTHORS



Prathima Mabel John     is Assistant Professor at Dayananda Sagar College of Engineering, Visvesvaraya Technological 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: prathimamabelise@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: ramamohanbabuise@dayanandasagar.edu