LETTER

HTTP DDoS Flooding Attack Mitigation in Software-Defined Networking

Sungho PARK[†a)], ***Member***, **Youngjun KIM**[†], **Hyungoo CHOI**[†], **Yeunwoong KYUNG**[††], ***and* Jinwoo PARK**[†], ***Nonmembers***

SUMMARY HTTP Distributed Denial of Service (DDoS) flooding attack aims to deplete the connection resources of a targeted web server by transmitting a massive amount of HTTP request packets using botnets. This type of attack seriously deteriorates the service quality of the web server by tying up its connection resources and uselessly holds up lots of network resources like link capacity and switching capability. This paper proposes a defense method for mitigating HTTP DDoS flooding attack based on software-defined networking (SDN). It is demonstrated in this paper that the proposed method can effectively defend the web server and preserve network resources against HTTP DDoS flooding attacks.

key words: *DDoS, HTTP flooding, network security, SDN*

1. Introduction

Distributed Denial of Service (DDoS) attack is a form of cyberattack in which malicious perpetrators neutralize a target's services by depleting network and connection resources through distributed attacks. Two primary types of DDoS attack are those that go through the network layer and those that utilize the application layer, with the latter the most common method of HTTP DDoS cyberattack. In an HTTP DDoS flooding attack, the perpetrator utilizes botnets to simultaneously transmit a massive number of HTTP request packets to overwhelm the target web server by depleting its resources, thus preventing it from offering its normal web services [1], [2].

There are various defense methods against the HTTP DDoS flooding attacks and these can generally be divided into two categories: destination-based defense method in which the target of the cyberattack initiates countermeasures, and network-based defense method in which the attacker's traffic is blocked at the network layer. Destination-based defense typically involves the web server blocking attacks by analyzing users' patterns and differentiating attackers from legitimate clients. A sophisticated form of the destination-based defense for a web server is CAPTCHA test, which is used to identify legitimate users from DDoS bots by asking users to identify and report back complex patterns of letters or distorted/layered characters [3]. Using this strategy, the botnets used in DDoS attacks can be identified. However, the destination-based countermeasures are initiated by target web servers after receiving attack traffic, so this could not be an appropriate defense system when dealing with HTTP DDoS flooding attacks.

Network-based defense utilizes network switches and routers to identify abnormal traffic patterns and initiate cyberattack countermeasures. As such, the network-based defense is preferred because countermeasures are deployed before the attack reaches the web servers. However, network-based defense is not widely used due to difficulties in accurately identifying and blocking attack traffic within the network.

As a response to this, the network-based defense methods utilizing software-defined networking (SDN) have been investigated [4]. Since a centralized SDN controller acquires a global view of the network, it can accurately identify attacks and manipulate the network switches and routers to efficiently target incoming attack traffic. The flow control scheme for DDoS defense called FlowFence was proposed by A. F. M. Piedrahita et.al. [5], in which network switches monitor the bandwidth usage and congestion condition. When congestion occur at a switch, it is notified to SDN controller. The controller identifies which flow carries an excessive amount of traffic and then sends a flow rule to switches to limit the bandwidth usage of the flow. S. Shin et.al. proposed an extensive SDN architecture called Avant-guard to detect and mitigate SYN Flood attacks [6]. The attacks are detected by monitoring flow dynamics at the data plane. When an attack event is detected, the SDN controller changes a flow table to mitigate the attack. In addition, the concentration of traffic and the increased processing load due to a heavy reliance on the SDN controller have been noticed and investigated to alleviate the problem [7], [8]. While these approaches leverage better network visibility from the SDN architecture, they still lack the analytic capabilities of destination-based defense to understand users' pattern and differentiating attackers from legitimate clients. Failing to leverage semantic information available at destinations, network-based defense ends up replying on rather simple attack identification and prevention mechanisms.

K. Hong et.al. is first in line to propose a mechanism that utilize the advantages of both destination-based defense (i.e. semantical analyzability at destinations) and network-based defense (i.e. early attack stop close to its source in the network). His slow HTTP DDoS attack defense scheme [9] cooperates a SDN controller cooperated with web servers to

Manuscript received March 9, 2021.
Manuscript publicized June 4, 2021.

[†]The authors are with Department of Electrical Engineering, Korea University, Seoul 02841, South Korea.
[††]The author is with Division of Computer Engineering, Hanshin University, Osan 18101, Gyeonggi-do, Korea.
a) E-mail: kuyt@korea.ac.kr

DOI: 10.1587/transinf.2021EDL8022

detect slow HTTP DDoS attack precisely at web servers and blocks the attack traffic early in the network.

Our idea is along the same line of philosophy as [9] and extends it to be able to handle heavy-traffic attacks such as HTTP DDoS flooding attack. The proposed defense method involves the SDN controller detecting HTTP DDoS flooding attack and blocking malicious attack traffic in the network layer and defending the malicious traffic at the attackers' side. This countermeasure not only identifies attacks carefully with the semantic-full information available at the web server but also eliminates the traffic caused by the DDoS flooding attacks on the network. The proposed method can detect a HTTP DDoS flooding attack more accurately and defend the attack less fatal to a web server and network because the proposed method utilizes web server to sensing suspicious user and makes edge switches to discard attackers' traffic.

2. Proposed HTTP DDoS Flooding Attack Mitigation Method

The configuration of the proposed HTTP DDoS flooding attack mitigation method based on SDN is illustrated in Fig. 1. The SDN-based network comprises a web server and a group of clients utilizing the web server. The web server is connected to the network through switch S_W, and the clients are connected to network through switch S_C. Switches S_W and S_C are assumed to be OpenFlow switches [10], while all clients are assumed to be connected through TCP to communicate with the web server. The clients consist of legitimate clients and attackers designated as C_i $(i = 1, 2, 3, \cdots)$. An application called HTTP DDoS Flooding Detector (HDFD) is assumed installed on the SDN controller, which is supposed to determine whether the HTTP requests delivered from a web server are for DDoS flooding attack.

The web server monitors the HTTP requests transmitted by the clients in order to identify any that are suspected to be HTTP DDoS flooding attack requests. This can be achieved by a way of comparing the number of HTTP request packets per unit time R_i from client C_i with the threshold value R_{th} at the web server. If $R_i > R_{th}$, the web server designates client C_i as a potential threat and stop processing HTTP requests from this client; rather it begins to forward the HTTP requests from C_i to the SDN controller as depicted in Fig. 2. The SDN controller then utilizes the IP address (IP_{Ci}) and the TCP Source Port number (P_{Ci}) in the HTTP request packets sent by C_i to adjust the flow rules for the OpenFlow switches S_C and S_W (Table 1). Using this switch control, the suspicious HTTP requests are redirected from the web server to the SDN controller, protecting the web server from HTTP DDoS flooding attacks.

While the SDN controller receives HTTP requests on behalf of the web server, the SDN controller transmits spoofed TCP ACK to the suspicious C_i instead of the web server to spoof normal transmission. This is intended to prevent the attacker from recognizing the countermeasures and thus changing their methods of attack. It also helps the SDN controller recover the web service quickly when the suspicious client turns out to be legitimate.

How to implement HDFD is beyond the scope of this paper, so it is not addressed here. However, for completion of the paper, we employed CAPTHA test as an example of HDFD as shown in Fig. 3 [3]. In order to find out whether C_i is an attacker, HDFD sends HTTP response containing CAPTCHA test to the suspicious C_i in response to HTTP request. If C_i is a legitimate client, C_i will submit a correct answer to the CAPTCHA test. If not, C_i can't respond to the CAPTCHA test appropriately. By doing so, HDFD can determine whether C_i is an attacker. When client C_i is identified as an attacker by HDFD, the SDN controller directs S_C to change the flow rule to block the HTTP request traffic sent from C_i toward the web server. Concurrently, the SDN

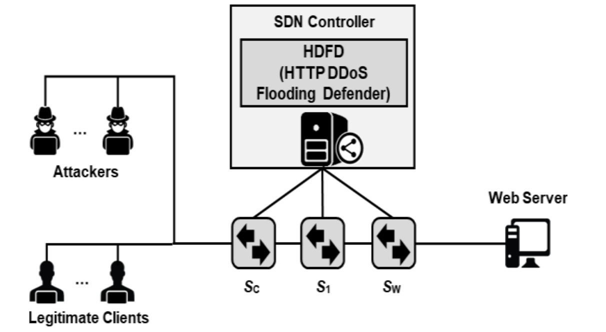

Fig. 1 SDN architecture for HTTP DDoS flooding attack mitigation

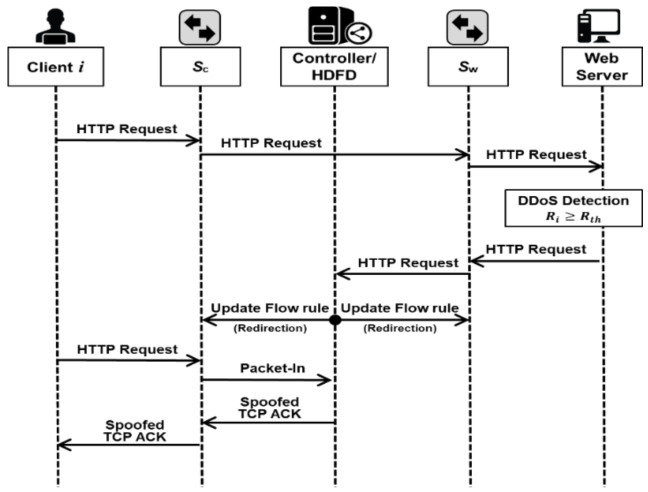

Fig. 2 Detection and redirection of HTTP DDoS flooding attack traffic

Table 1 Flow rules for S_C and S_W

	Match Field				Action Field
	IP_{src}	IP_{dst}	TCP_{src}	TCP_{dst}	Action
S_C	IP_{Ci}	IP_W	P_{Ci}	80	PACKET_IN
S_W	IP_W	IP_{Ci}	80	P_{Ci}	PACKET_IN

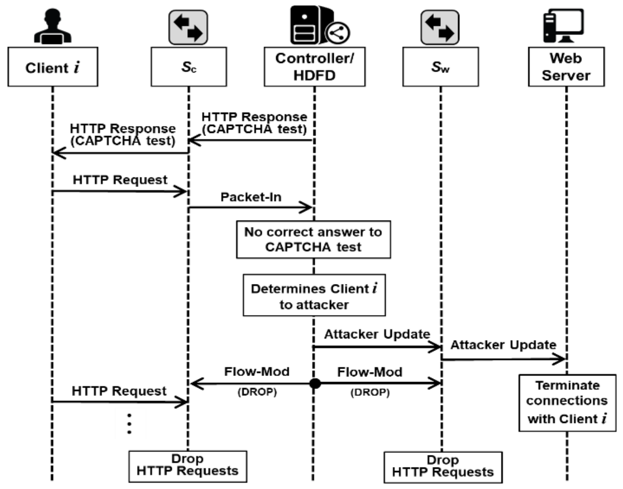

Fig. 3 Blocking the HTTP DDoS flooding attack traffic

controller transmits the detection result with C_i-related information to the web server to ensure the termination of the connection with C_i. It helps the web server save the connection resources as illustrated in Fig. 3. When C_i turns out to be a legitimate user, the TCP connection between the client and the web server is restored. To accomplish this, the SDN controller directs S_W and S_C to delete the flow rules previously modified to redirect the client's traffic to the SDN controller. It restores the normal services from the web server for C_i.

3. Performance Evaluation

To validate the performance of the proposed HTTP DDoS flooding defense method, a network model was designed by utilizing the Mininet simulation program [11]. In establishing the web server parameters, Apache HTTP Server which is most commonly used server on the internet, was referenced (Table 2) [12]. In order to complete the performance evaluation of the proposed defense method, CAPTCHA test in employed for HDFD [3].

As shown in Fig. 4, the attackers initiate the HTTP DDoS flooding attacks after 5 seconds, leading to a rapid increase in the number of HTTP connections to the web server. With the initiation of the attack, all 256 connections to the web server are fully occupied, which is very likely to cause serious service disruption at the web server. Figure 5 presents the results when the HDFD countermeasures are employed on the same HTTP DDoS flooding attack. When the attack is initiated, the number of occupied HTTP connections for the web server rapidly increases. However, the interaction between the web server, the SDN controller, and the HDFD quickly restores the number of the HTTP connections to a normal level. The time required for this recovery is approximately 2.3 seconds in our simulation, which includes 0.4 seconds for the web server to identify the suspicious HTTP requests, 0.2 seconds for the web server to transmit the suspicious HTTP request packets to the SDN controller, 1.3 seconds for the SDN controller to identify the suspicious requests as an actual attack, 0.2 seconds for the SDN controller to inform the network switches and routers, and 0.2 seconds for the switches and routers to block the malicious HTTP requests.

Figure 6 presents the variation in the total number of HTTP requests passing through network switch S_C when the HDFD is active. It can be noticed that the HDFD countermeasure can effectively restore the number of HTTP requests, which was dramatically increased due to an HTTP DDoS flooding attack, to a normal level. This verifies that the proposed defense method not only protects the web server from HTTP DDoS flooding attacks but also has abil-

Table 2 Simulation parameters

Parameter		Value
Web Server	Maximum connections	256
	R_{th}	10
	Response time	0.5 (s)
HDFD	N_{th}	50
Attacker	Number of attackers	40
	Number of requests per interval	50
	Attack interval	1 (s)
	Attack start time	5 (s)
Legitimate Client	Number of legitimate clients	100
	Average request rate	1 (/s)

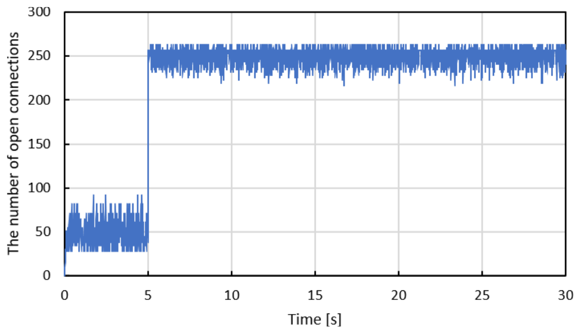

Fig. 4 The number of occupied connections of web server by HTTP DDoS flooding attack

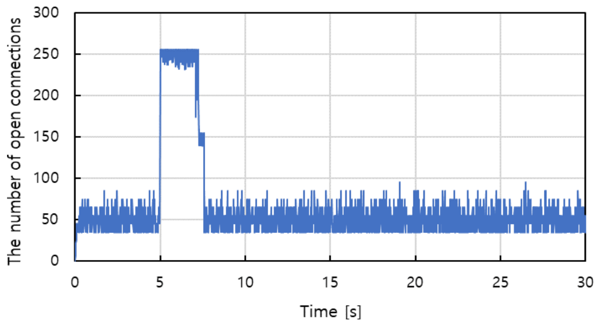

Fig. 5 The improved number of occupied connections of web server when HDFD is activated

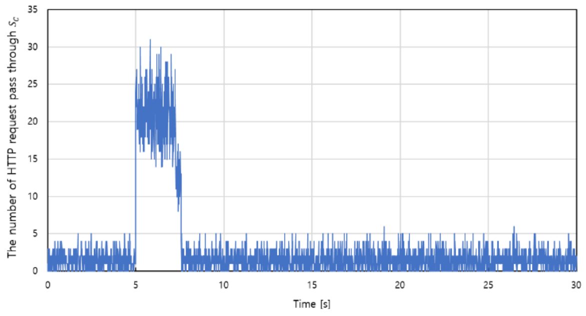

Fig. 6 Variation of the number of HTTP requests passing through S_C when HDFD is activated

ity to resolve the significant waste of network resources due to DDoS attack traffic.

4. Conclusion

In this paper, we propose a HTTP DDoS flooding defense method based on SDN that can effectively protect the target web server from HTTP DDoS flooding attack. The SDN controller with HDFD intercepts the HTTP DDoS flooding traffic transmitted by suspicious clients on behalf of the targeted web server, and analyzes the traffic to find out whether the clients are real attackers. When any client is decided as an attacker by HDFD, the SDN controller blocks the HTTP DDoS flooding traffic from the client thru manipulating switches in the network layer. It was demonstrated in this paper that the proposed defense method can protect the web server and resolve the network resource depletion problem caused by HTTP DDoS flooding attacks.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2020R1G1A1100493).

References

[1] S.T. Zargar, J. Joshi, and D. Tipper, "A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks," IEEE Communications Surveys & Tutorials, vol.15, no.4, pp.2046–2069, 2013.
[2] S. Dong, K. Abbas, and R. Jain, "A Survey on Distributed Denial of Service (DDoS) Attacks in SDN and Cloud Computing Environments," IEEE Access, vol.7, pp.80813–80828, 2019.
[3] M. Mehra, M. Agarwal, R. Pawar, and D. Shah, "Mitigating denial of service attack using CAPTCHA mechanism," ICWET Workshop on Emerging Trends in Technology, ACM New York, NY, USA, pp.284–287, 2011.
[4] Q. Yan, F.R. Yu, Q. Gong, and J. Li, "Software-Defined Networking (SDN) and Distributed Denial of Service (DDoS) Attacks in Cloud Computing Environments: A Survey, Some Research Issues, and Challenges," IEEE Communications Surveys & Tutorials, vol.18, no.1, pp.602–622, 2016.
[5] A.F.M. Piedrahita, S. Rueda, D.M.F. Mattos, and O.C.M.B. Duarte, "FlowFence: A denial of service defense system for software defined networking," Proc. Global Inf. Infrastruct. Netw. Symp. (GIIS), pp.1–6, Oct. 2015.
[6] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Avant-guard: Scalable and vigilant switch flow management in software-defined networks," Proc. ACM SIGSAC Conf. Comput. Commun. Secur., pp.413–424, 2013.
[7] J. Yang, X. Yang, Z. Zhou, X. Wu, T. Benson, and C. Hu, "Focus: Function offloading from a controller to utilize switch power," Proc. IEEE NFV-SDN, 2016.
[8] D. Gao, Z. Liu, Y. Liu, C.H. Foh, T. Zhi, and H.-C. Chao, "Defending against Packet-In messages flooding attack under SDN context," Soft Computing, vol.22, pp.6797–6809, 2018.
[9] K. Hong, Y. Kim, H. Choi, and J. Park, "SDN-Assisted Slow HTTP DDoS Attack Defense Method," IEEE Communications Letters, vol.22, no.4, pp.688–691, 2018.
[10] OpenFlow Switch Specification Version 1.5.1, Open Networking Foundation, https://www.opennetworking.org
[11] Mininet, http://mininet.org
[12] Apache HTTP server, https://httpd.apache.org