A Method for Preventing Slow HTTP DoS attacks

Koichi Ozaki[1)], Astushi Kanai[2)]
Faculty of Science and Technology
Hosei University
Tokyo, Japan
[1)] koichi.ozaki.3t@stu.hosei.ac.jp, [2)]yoikana@hosei.ac.jp

Shigeaki Tanimoto
Faculty of Social Systems Science
Chiba Institute of Technology
Chiba, Japan
shigeaki.tanimoto@it-chiba.ac.jp

***Abstract*—A Slow Hypertext-Transfer-Protocol (HTTP) Denial-of-service (DoS) Attack looks like a genuine user and can block access to genuine users. Over the past few years, several studies have been performed on the defense against Slow HTTP DoS Attacks. However, little attention has been given to a Slow HTTP DoS Attack that resembles a normal DoS Attack. In this paper, the effectiveness of setting the longest session time and the longest packet interval with an appropriate threshold was evaluated by changing each threshold and comparing the results. As a result, we demonstrated the effectiveness of the proposed method. To prevent a Slow HTTP DoS attack completely, it is necessary to not only take measures for typical Slow HTTP DoS attacks but also set a threshold for anomaly detection in consideration of Slow HTTP DoS attacks that resemble a normal DoS attack.**

Keywords- Slow HTTP DoS Attack; session time; packet interval

I. INTRODUCTION

DoS attacks are mainly classified as three types of attacks [1]. The first type is an attack that sends mass requests or a huge amount of data to a leased line and thereby fills up the line’s bandwidth. The second type is an attack that exhausts the system resources (processing capacity of central-processing-units (CPUs), memory, etc.) of a Web server. The third type is an attack that exploits vulnerabilities of routers and servers. The aims of these attacks are to violate the availability of services and to impose the accompanying economic burden on the server owner. If a DoS attack is considered from the viewpoint of the layers of the network system, when the DoS attacks were initially made, the network layer and the transport layer were often attacked with a large amount of data traffic. However, as DoS attacks diversified over the years, they started to attack the application layer with a small amount of data traffic. Most DoS Attacks targeting the application layer are difficult to detect because many of them follow regular processes in the network layer and the transport layer. A “Slow HTTP DoS attack” is one such attack targeting the application layer [2][3]. Unlike other DoS attacks, as shown in Figure 1, it continues Transmission-Control-Protocol (TCP) sessions for a long time with a small number of packets. A normal communication and a Slow HTTP DoS attack are shown in Figure 2.

The attack method is classified into three categories: “Slow HTTP Headers Attack,” “Slow HTTP BODY Attack,” and “Slow Read DoS Attack,” depending on how the duration of the TCP session is extended. A Slow HTTP Headers Attack (aka “Slowloris”) extends the duration of a TCP session by sending a long HTTP request header little by little with a wait time in between returning responses and sending requests. A Slow HTTP BODY Attack (aka “Slow HTTP BODY Attack” or “R.U.D.Y”) extends the duration of a TCP session by sending a long HTTP request body little by little with a waiting time in between returning responses and sending requests. A Slow Read DoS Attack extends the duration of a TCP session by specifying a very small TCP window size and receiving an HTTP response from the Web server little by little. This rest of paper is organized as follows: Section Ⅱ introduces related works, Section Ⅲ describes the proposed method for prevent Slow HTTP DoS Attacks, Section Ⅳ describes the experimental environment under which the method was evaluated, Section Ⅴ presents results of an evaluation of the effectiveness of the method, and Section Ⅵ presents the conclusions of this work.

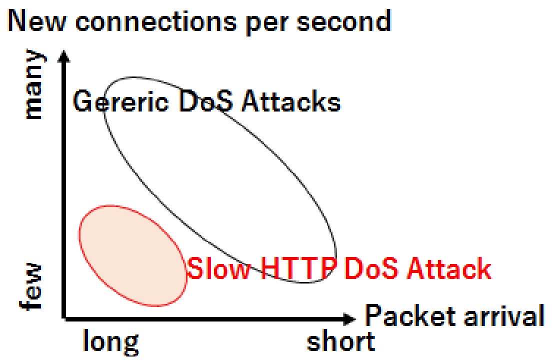

Figure 1. Conceptual diagram of the range of a Slow HTTP DoS Attack.

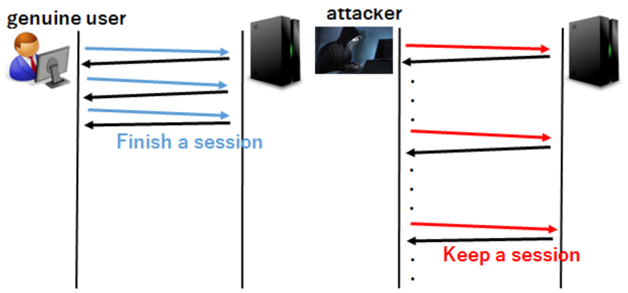

Figure 2. Normal communication and Slow HTTP DoS Attack

II. RELATED WORKS

Generic DoS attacks with large amounts of data traffic can be detected by anomaly detections and signature detections. However, a Slow HTTP DoS Attack looks like a genuine user, and it can attack the Web server (without alerting the Web server) with a small amount of traffic. Accordingly, it cannot be detected by anomaly detections; it can only be detected by signature detections. Over the past few years, how to defend against a Slow HTTP DoS Attack has been studied [3]-[7]. However, many problems remain to be solved. For example, a method of limiting the number of simultaneous sessions from the same Internet-Protocol (IP) address has been introduced [8]. However, when multiple genuine users use a common Network-Address-Translation (NAT) and simultaneously use a Web server with the same global address, the Web server may recognize genuine users as attackers and restrict their accesses. Also, if the attacker imitates an IP address, uses multiple IP addresses, or uses a Botnet, the defense method cannot defend the Web server as shown in Figure 3 [9].

Another method of defense is to limit parameters such as longest session time, minimum reception rate, and longest packet interval [10]. However, a genuine user communication via a Secure-Socket-Layer (SSL) or slow communication lines must not be misrecognized as an attacker. Also, Slow HTTP DoS Attacks have received little attention compared to that paid to normal DoS Attacks. Even though it is configured to detect only typical Slow HTTP DoS Attacks, the defense based on this method cannot defend Web servers from a Slow HTTP DoS Attack that resembles a normal DoS attack in order to sneak through the detection mechanism.

A so-called high-performance “Web-application firewall” (WAF) compares an assumed amount of data with the actual amount of data while gradually decreasing window sizes. It thereby distinguishes genuine users from attackers [11]. However, a high-performance WAF is costly, and in some cases, it cannot be introduced from the viewpoint of the balance between asset value and risk of service outage. High-performance protection with low cost and easy set-up is thus desired.

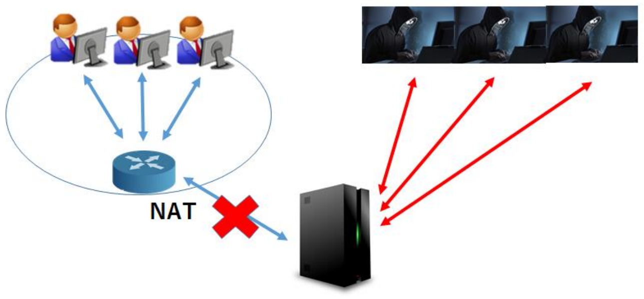

Figure 3. Problems when limiting access by IP address

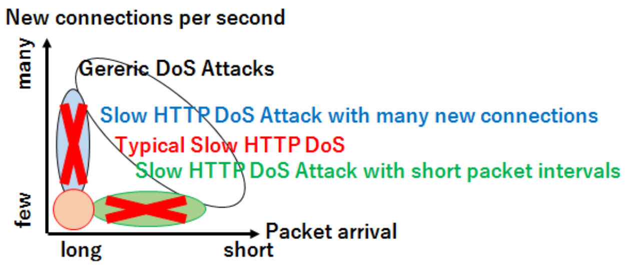
New connections per second
many
few
Gereric DoS Attacks
Slow HTTP DoS Attack with many new connections
Typical Slow HTTP DoS
Slow HTTP DoS Attack with short packet intervals
Packet arrival
long
short

Figure 4. Position of Slow HTTP DoS Attacks in relation to generic DoS

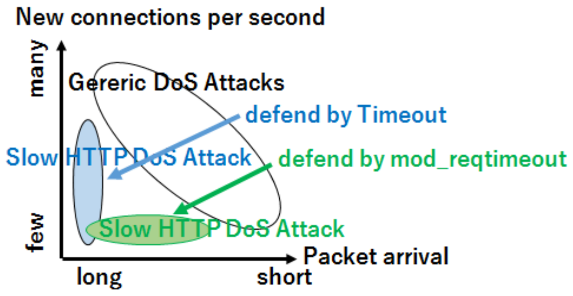

Figure 5. Conceptual diagram of the range to be defended

III. PROPOSED APPROACH

It is relatively easy to detect typical Slow HTTP DoS Attacks with long packet intervals and little conections. However, attackers may sometimes make a Slow HTTP DoS Attack like a normal DoS attack in order to sneak through a detection-and-defense mechanism. Little attention has been paid to such attacks. It is impossible to prevent such attacks if, as shown in Figure 4, the threshold length of the longest session time and the longest packet interval are not appropriate or only one defense measure is applied. A defense method proposed in this study limits session time, packet interval and average reception rate with appropriate values. As shown in Figure 5, it can thus prevent a wider range of Slow HTTP DoS Attacks.

The Web service was unavailable when the number of connections exceeds the-maximum number-of-connections-that-could-access-the-Web-server (Maxclients). In this proposed method, it is whether the packet is for an attack or a usual usage in following three steps. In step 1, when the average packet interval is longer than the threshold of packet intervals, it is judged as an attack. Thereby, if the number of connections connected within the packet interval threshold time does not exceed Maxclients, the Web service becomes available. However, even if the packet intervals are limited, attacks with short packet intervals cannot be blocked. Such attacks are prevented in step 2 and step 3. Step 2 prevents false detection of a usual usage who takes much traffic and long communication time as an attack. If the average reception rate is larger than the threshold of reception rate, it is judged as a usual usage. Otherwise, the process shifts to step 3. In step 3, when the session time is longer than the threshold of session time, it is judged to be an attack.

Thereby, if the number of connections connected within the session time threshold time does not exceed Maxclients, the Web service becomes available.

IV. EXPERIMENTAL ENVIRONMENT

An experimental environment in which a defending Web server and an attacking client are directly connected by a switch was set up as shown in Figure 6.

A. *Environment of the defending Web server*

The OS of the defending Web server used CentOS 6.5, and Apache version 2.2.27 (with mod_reqtimeout as a standard feature) [12][13]. The Apache configuration was set in the /etc/httpd/conf/httpd.conf file, and the main configuration is listed in TABLE Ⅰ. The maximum number of connections that could access the Web server was 256. The longest packet interval was limited by setting the value of Timeout to 2 or 60 s. The mod_reqtimeout configuration of the defending server was described in httpd.conf. The longest session time was limited by setting the value of mod_reqtimeout to 20 or 3 s. When the average reception rate was 300 Mbps or more, the time limit was extended to 120 s.

B. *Environment of the attacking client*

The attacking client's OS used Ubuntu 14.04, and slowhttptest 1.7 was used as an attack-testing tool [14][15]. The purpose of the attacking client is usually to occupy all connections with a small amount of traffic so that the defending Web server does not notice it is being attacked. As such a typical Slow DoS HTTP Attack, the attacking client attacked with 15 new connections per second and with a packet interval of 10 s. Also, for attacks with short packet intervals, the attacking client made a Slow HTTP DoS Attack with 15 new connections per second and short packet intervals of 1 s. Moreover, for attacks with many new connections per second, the attacking client made a Slow HTTP DoS Attack with 100 new connections per second and a packet interval of 10 s. Both Slow HTTP Headers Attacks and Slow HTTP BODY Attacks were made, and the experimental results were evaluated. Both attacks were set as common configurations as shown in TABLE Ⅱ. The experimental result was evaluated by the HTML generated by the attacking client's slowhttptest.

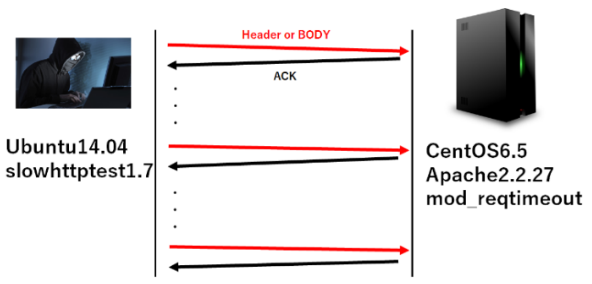

Figure 6. Experimental environment

TABLE I. APACHE CONFIGURATION

Timeout	60 or 2
KeepAlive	On
MaxKeepAliveRequests	5
KeepAliveTimeout	2
StartServers	8
MinspareSevers	5
MinspareSevers	20
ServerLimit	256
MaxClients	256
MaxRequestsPerChild	4000

TABLE II. COMMON CONFIGURATION OF SLOW HTTP HEADERS ATTACK AND SLOW HTTP BODY ATTACK

Total number of connections	300 or 2000
Number of new connections per second	15 or 100
Should results be generated in CSV and HTML format	Yes
Path and name of generated file	for example：head-test1
Response RTT to check connection status (s)	1
Attacked URL	http://centostestsrv.com
Test time (s)	20
Packet interval (s)	10 or 1

V. EVALUATION

In this paper, implementation and evaluation are not as Section Ⅲ, but based on the following test model in two steps. In step 1, packet interval is longer than the threshold of packet intervals, it is judged as an attack. And the effectiveness of appropriately limiting the packet intervals was evaluated by changing the threshold of the longest packet interval and comparing the results. The inappropriate threshold was set to 60 s (which has been used by default). The appreciate threshold was set to two seconds in consideration of genuine users who are communicating via SSL or a slow communication line. In step 2, when the session time is longer than the threshold of session time, it is judged to be an attack. And the effectiveness of appropriately limiting the session time was evaluated by changing the threshold of the longest session time and comparing the results. The inappropriate threshold was set to 20 s (which has been conventionally used). The appropriate threshold was set to 3 seconds in consideration of a genuine user using communication via SSL or a slow communication line.

This paper focuses only on Slow HTTP Headers Attacks and Slow HTTP Body Attacks, not Slow HTTP Read Attacks. Effectiveness of the proposed attack-prevention method was experimentally evaluated under four conditions, namely, “Timeout,” “mod_reqtimeout” setting of the defending server, “packet interval,” and “number of new connections per second” of the attacking client, listed as “cases A to D” in Table Ⅲ.

A. *Typical Slow HTTP DoS Attack (case A)*

Timeout of the defending Web server was set to 60 s as the default setting, and the threshold of mod_reqtimeout was set to 20 s. The attacking client made a typical Slow HTTP

DoS Attack with 15 new connections per seconds and a packet interval of 10 s. The experimental results when the Slow HTTP Headers Attack was made and those when the Slow HTTP BODY Attack was made are shown in Figures 7 and 8, respectively.

As for the graphs in the figures, the horizontal axis shows the elapsed time of the experiment, the blue line on the vertical axis indicates the number of closed connections, the red line indicates the number of waiting connections, the yellow line indicates the number of connections being made, and the green line indicates whether the Web server service is available or not.

As shown in the figures, the Web service became unavailable because the number of connections established during the longest session time exceeded MaxClients. Even the typical Slow HTTP DoS Attack could not be prevented because the longest session time was limited inappropriately.

TABLE III. VALIDATION CONTENTS

case	Timeout (s)	mod_reqtimeout (s)	packet interval (s)	number of new connections/s
A	60	20	10	15
B	60	3	1	15
C	60	3	10	100
D	2	3	10	100

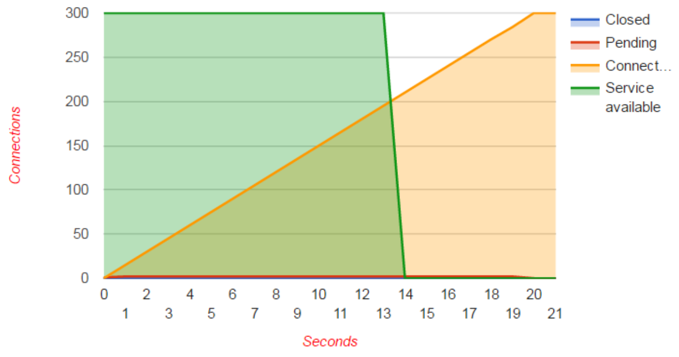

Figure 7. Typical Slow HTTP Headers Attack on Web server with incorrect mod_reqtimeout

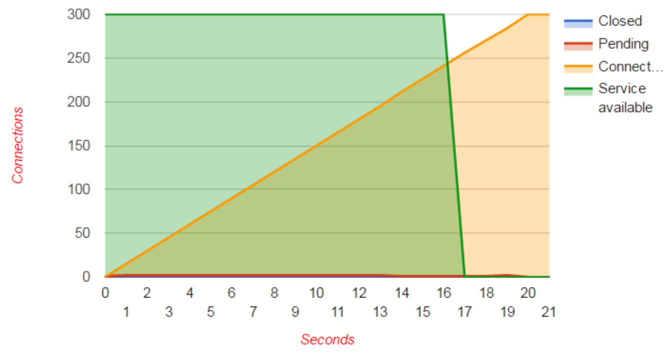

Figure 8. Typical Slow HTTP BODY Attack on Web server with incorrect mod_reqtimeout

B. Slow HTTP DoS Attack with short packet intervals (case B)

Timeout of the defending Web server was set to 60 s (as the default setting value), and the threshold of mod_reqtimeout was set to 3 s. The attacking client made a Slow HTTP DoS Attack with 15 new connections per second and a short packet interval of 1 s. The experimental results when the Slow HTTP Headers Attack was made and when the Slow HTTP BODY Attack was made are shown in Figures 9 and 10, respectively.

As shown in the figures, the Web service was available because the number of connecting connections was stable at 70 to 80, and the connections were closed steadily. The Slow HTTP DoS Attack with short packet intervals could be prevented because the longest session time was limited appropriately.

C. Slow HTTP DoS Attack with many new connections per second (case C)

Timeout of the defending Web server was set to 60 s as the default setting value, and the threshold of mod_reqtimeout was set to 3 s. The attacking client made a Slow HTTP DoS Attack with many (100) new connections per second and a packet interval of 10 s. The results when the Slow HTTP Headers Attack was made and when the Slow HTTP BODY Attack was made are shown in Figures 11 and 12, respectively.

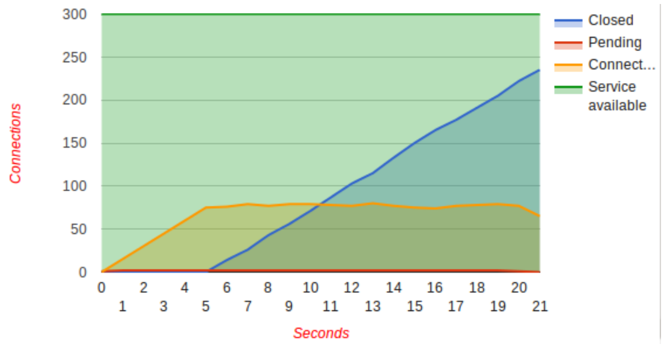

Figure 9. Slow HTTP Headers Attack with short packet intervals on Web server with appropriate mod_reqtimeout

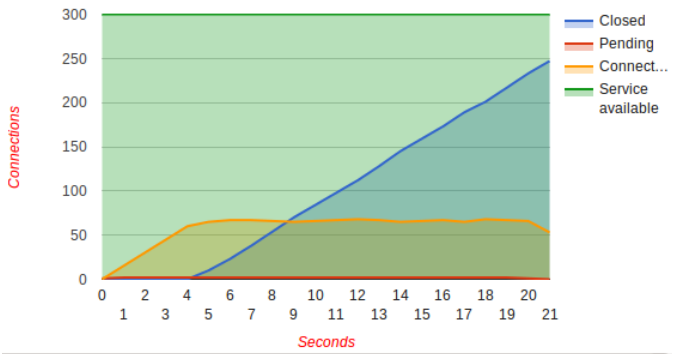

Figure 10. Slow HTTP BODY Attack with short packet interval on Web server with appropriate mod_reqtimeout

As shown in the figures, the service became unavailable because the number of new connections being made was larger than the number of connections closed. Even though the longest session time limit is appropriate, a Slow HTTP DoS Attack with many new connections could not be prevented.

D. *Slow HTTP DoS attack with many new connections per second (case D)*

Timeout of the defending Web server was set to 2 s, and the threshold of mod_reqtimeout was set to 3 s. The attacking client made a Slow HTTP DoS Attack with many (100) new connections per second, and a packet interval of 10 s. The results when a Slow HTTP Headers Attack was made and when a Slow HTTP BODY Attack was made are shown in Figures 13 and 14, respectively.

As shown in the figures, the Web service was available because the number of connections being made was stable (except for a short time) below MaxClients of 256. However, when it exceeded MaxClients for only the short time, the service was unavailable. This instability is considered to be due to processing delay of Apache and mod_reqtimeout. The Slow HTTP DoS attack with many new connections could be prevented because the longest packet interval was limited appropriately.

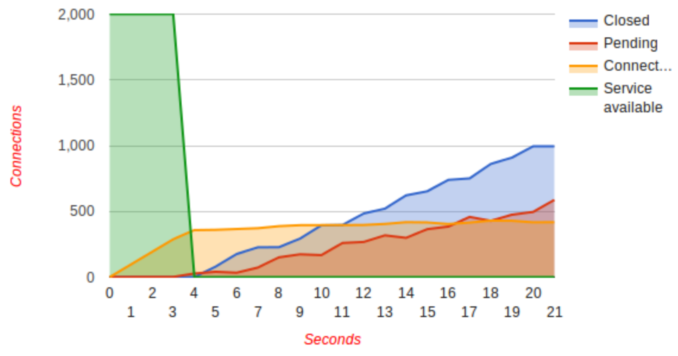

Figure 11. Slow HTTP Headers Attack with many new connections per second on Web server with appropriate mod_reqtimeout

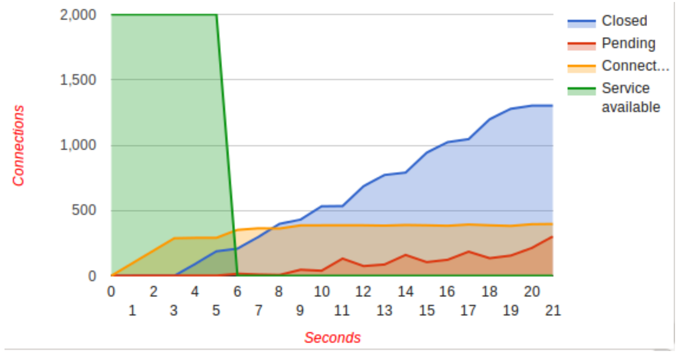

Figure 12. Slow HTTP BODY Attack with many new connections per second on Web server with appropriate mod_reqtimeout

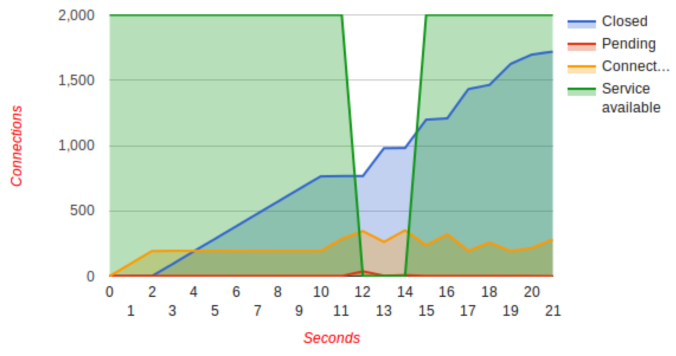

Figure 13. Slow HTTP Headers Attack with many new connections per second on Web server with appropriate Timeout

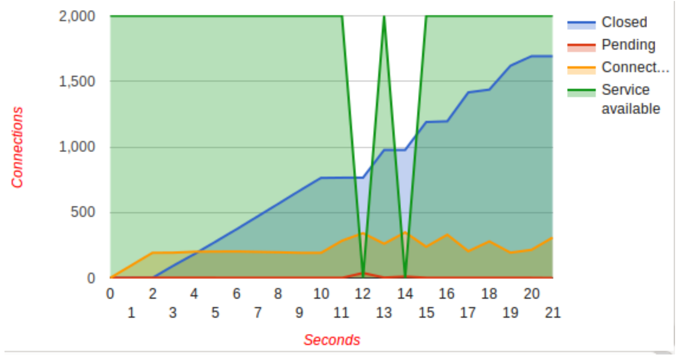

Figure 14. Slow HTTP BODY Attack with many new connections per second on Web server with appropriate Timeout

VI. CONCLUSION

In this experiment, aiming to sneak through detection by a defending Web server against a Slow HTTP DoS Attack, the attacking client made an attack with many new connections per second (with 100 new connections per seconds and a packet interval of 10 s) or an attack with short packet intervals (with 15 new connection per seconds and packet a packet interval of 1 s). These attacks could be prevented by limiting the longest packet interval and longest session time. In other words, applying multiple measures with an appropriate threshold was effective in preventing these attacks. However, this defense method cannot prevent attacks in which the number of new connections per second is further increased and "Timeout × new connections per second > MaxClient" (example: an attack with 150 new connections per seconds and second packet interval of 10 s) or an attack with many new connections per second and short packet intervals (example: an attack with 100 new connections per seconds and second packet interval of 1 s). However, increasing the number of new connections per second or shortening the interval between packets means increasing the number of packets. Such attacks with such a large number of packets are subject to anomaly detection against general DoS attacks intended to fill the line bandwidth. To prevent a Slow HTTP DoS Attack completely, it is necessary to not only take measures for typical Slow

HTTP DoS Attacks but also set a threshold for anomaly detection in consideration of Slow HTTP DoS Attacks that resemble a normal DoS Attack.

The appropriate Timeout and mod_reqtimeout thresholds will change depending on the service provided by the Web server, communication method, and so on. If a genuine user accesses the Web server with the defense method in this study via SSL or a line with low communication speed, and communication takes time due to sending of large files, they may be misrecognized as an attacker. In this evaluation, two kinds of the threshold of packet interval and session time was set and evaluated, but it was not the best threshold. Also, there was no setting of the threshold of the minimum reception rate. Accordingly, a future direction of this study will evaluate all the threshold in detail and reduce the possibility of misrecognizing a genuine user as an attacker as much as possible and expand the range that can be defended by further improving the detection accuracy and performance of the proposed method for preventing Slow HTTP DoS Attacks.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Number 15H02783.

REFERENCES

[1] AndMen, “About DoS/DDoS Attack,” [online]. Available: http://andmem.blogspot.jp/2014/02/dosattack.html. [retrieved: 7, 2017].

[2] E. Cambiaso, G. Papaleo, G. Chiola, and M. Aiello, “Slow DoS attacks: definition and categorisation,” Int. J. Trust Management in Computing and Communications, Volume 1, Number 3-4, pp.300-319, 2013.

[3] @police, “Notice on Slow HTTP DoS Attack,” [online]. Available: https://www.npa.go.jp/cyberpolice/detect/pdf/20151216.pdf. [retrieved: 1, 2017]

[4] Kuzmanovic and E. Knightly, “Low-Rate TCP -Targeted Denial of Service Attacks (The Shrew vs. the Mice and Elephants),” roceedings of ACM SIGCOMM 2003, Karlsruhe, Germany, August 2003, pp. 75-86.

[5] Dalia Nashat, Xiaohong Jiang, and Susumu Horiguchi, “Router based detection for Low-rate agents of DDoS attack,” In 2008 International Conference on High Performance Switching and Routing, pp.177–182, May 2008.

[6] Amey Shevtekar and Nirwan Ansari, “A Proactive Test Based Differentiation Technique to Mitigate Low Rate DoS Attacks,” In 2007 16th International Conference on Computer Communications and Networks, August 2007.

[7] Ian Muscat, “How To Mitigate Slow HTTP DoS Attacks in Apache HTTP Server,” [online]. Available: https://www.acunetix.com/blog/articles/slow-http-dos-attacks-mitigate-apache-http-server/ [retrieved: 8, 2017].

[8] Jieren Cheng, Jianping Yin, Yun Liu, Zhiping Cai, and Min Li, “DDoS attack detection algorithm using IP address features,” In Frontiers in Algorithmics, pages 207–215. Springer, 2009.

[9] Esraa Alomari, Selvakumar Manickam, B. B. Gupta, Shankar Karuppayah, and Rafeef Alfaris, “Botnet-based Distributed Denial of Service (DDoS) Attacks on Web Servers,” Classification and Art. International Journal of Computer Applications, July 2012. Published by Foundation of Computer Science, New York, USA.

[10] S. Sarat and A. Terzis, “On the Effect of Router Buffer Sizes on Low-Rate Denial of Service Attacks,” Proceedings of IEEE ICCCN 05, San Diego, California, October 2005, pp.281-286.

[11] @IT, “Barracuda strengthens the WAF appliance, measures to "Slow DoS Attack",” [online]. Available: http://www.atmarkit.co.jp/ait/articles/1211/09/news067.html. [retrieved: 7, 2017].

[12] CentOS, “Download CentOS,” [online]. Available: https://www.centos.org/download/. [retrieved: 7, 2017].

[13] Apache, “Download - The Apache HTTP Server Project,” [online]. Available:https://httpd.apache.org/download.cgi. [retrieved: 7, 2017].

[14] Ubuntu, “The leading operating system for PCs, TABLEts, phones, IoT devices, servers and the cloud | Ubuntu,” [online]. Available:https://www.ubuntu.com. [retrieved: 7, 2017].

[15] slowhttptest, “GitHub - shekyan/slowhttptest: Application Layer DoS Attack simulator,” [online] Available:https://github.com/shekyan/slowhttptest. [retrieved: 7, 2017]