2016 19th International Conference on Network-Based Information Systems

A Defense Method against Distributed Slow HTTP DoS Attack

Tetsuya Hirakawa, Kanayo Ogura, Bhed Bahadur Bista, Toyoo Takata,
Faculty of Software and Information Science, Iwate Prefectural University
152-52, Sugo, Takizawa-shi, Iwate 020-0693, Japan
{g231o025, ogura_k, bbb, takata}@iwate-pu.ac.jp

***Abstract*—Threat of Distributed Denial of Service (DDoS) attack, that attempts to make a machine or network resource unavailable is getting serious. A service provided for load testing was abused to perform a DDoS attack targeting online game company in September, 2014. Such services are bringing serious threat for most services on the Internet. Slow HTTP DoS attack is one of the DoS attack methods that targets HTTP servers. This method obstructs the service by saturating the connection pool with slow and many requests.**

It is known that Slow HTTP DoS attack by just one attacker can be prevented effectively by limiting the number of connections for each IP address. On the other hand, it is also known that it is difficult to defend from Slow HTTP DoS attack from multiple attackers (i.e. Distributed Slow HTTP DoS attack).

The threat of DDoS attack is getting serious, so we need a effective defense method against Distributed Slow HTTP DoS attack. In this paper, we propose and evaluate a defense method against Distributed Slow HTTP DoS attack by disconnecting the attack connections selectively by focusing on the number of connections for each IP address and the duration time. We show in our research that our proposed defense method is effective against Distributed Slow HTTP DoS attack.

I. Introduction

In recent years, the threat of Distributed Denial of Service (DDoS) attack is getting serious.

The number of DDoS attacks in third quarter of 2015 increased by 179.66% in comparison with third quarter of 2014, and increased by 22.79% in comparison with second quarter of 2015[1].

One of the DDoS threat is a service called booter (also called stresser). Booter is a service that generates huge traffic for load testing. For example, Stress Boss[2] a booter service says that they can generate traffic of 240 Gbps in 300 seconds costing only $3.99. Therefore, attackers can cause huge scale DDoS attack easily abusing booters.

Vulnerable Internet of Things (IoT) devices are also bringing the threat. The IoT is a system that connects smart devices to the Internet. IoT devices tend to have the following characteristics[3]: Always power-on, fairly high band width, weak password, standards are lagging, update cycles are long. These characteristics make weak IoT devices easy to be hijacked and abused to launch a DDoS attack.

Slow HTTP DoS attack is one of the DoS attack methods that targets HTTP servers. This method obstructs the service by saturating the connection pool with slow and many requests. Park et al.[4] reported that Slow Read DoS attack, a kind of Slow HTTP DoS attack, from just one attacker can be prevented effectively by limiting the number of connections for each IP address. On the other hand, Park et al. also reported that it is difficult to defend against Slow HTTP DoS attack from multiple attackers (i.e. Distributed Slow HTTP DoS attack).

Booters and vulnerable IoT devices are bringing serious threat of DDoS attack today, therefore we need an effective defense method against Distributed Slow HTTP DoS attack.

II. Slow HTTP DoS Attack

Slow HTTP DoS attack is a DDoS method that saturates a Web server with many requests, as opposed to the attack to saturate a network such as the NTP reflection attack.

Currently, three types of Slow HTTP DoS attacks are known[5]: Slow HTTP Headers, Slow HTTP POST, and Slow Read Attack. These methods are common in that they interferes in legitimate client's connection by generating and keeping a large number of connections to saturate the connection pool of the server.

We will describe these methods below.

A. Slow HTTP Headers

Slow HTTP Headers is an attack method in which the attacker sends a large HTTP request headers slowly in order to keep the connection. This method was firstly implemented as a tool named Slowloris, so it is also called Slowloris attack. Only The Apache HTTP Server is affected by this attack. The server will disconnect if the request headers does not arrive within a certain period of time. Therefore the attacker will try to send a large request headers as slowly as possible to keep the connection for a long time. In Apache 2.4, the size of request headers is restricted by the `LimitRequestFieldSize` directive and the `LimitRequestFields` directive[6]. These directives configure the maximum byte count of one header field and the maximum number of header fields respectively. The default values are 8190 bytes and 100. Therefore, the attacker can send the header up to 819 KB (8190×100 bytes) to keep the connection.

A hacker group "Anonymous" introduced the Slowloris attack tool in a document they published in November 2015[7], that described the way to launch a cyber attack against the terrorist organization "Islamic State in Iraq and the Levant" (ISIL).

2157-0426/16 $31.00 © 2016 IEEE
DOI 10.1109/NBiS.2016.58

This method was announced in June 17th, 2009, therefore it is not a recent attack technique. However, it was still effective in 2015 as the Anonymous's recommendation.

B. Slow HTTP POST

Slow HTTP POST is an attack method in which the attacker sends a large data slowly as request body using the HTTP POST method to in order to keep the connection.

Like Slow HTTP Headers, larger request body a server accepts, easier it is for an attacker to keep the connection. In Apache 2.4, the size of request body is restricted by the `LimitRequestBody` directive. The default value is 0 (no limit).

C. Slow Read Attack

Slow Read attack is a method in which the attacker keeps the connection by receiving the response from the server slowly using very small TCP window size.

Contrary to the two methods described above, in which send some data slowly, this method forces the server to send data slowly.

The window size can be set to beyond 0 by the attacker arbitrarily. Smaller window size makes it easier to keep the connection by the attacker, but also makes it easier to detect the attack by the server. For example, slowhttptest[8] is a tool, that implements these three attack methods, set the window size to a random value from 8 to 16 bytes as the default.

This attack needs at least one content that is larger than the send buffer on the target server. The server will finish processing the request when the content has stored to the send buffer, so if the content is smaller than the send buffer, the attack will be failed. The size of the send buffer is usually from 65 to 128 KB[9].

On the other hand, it is ascertained that the average size of web pages reached to 965 KB in December 2011 and it is still increasing[10]. This is enough size to succeed in Slow Read Attack.

III. Previous works

A. General measures against DoS

We will introduce general measure against (D)DoS attack, not only against Slow HTTP DoS attack.

1) Darknet observation: Darknets are "a portion of routed, allocated IP space in which no active services or servers reside."[11] In other words, no normal packet will arrive to the darknet. Packets that arrive to the darknet are caused by malicious action, such as activities of malwares, backscatters of DDoS attack, otherwise human errors (e.g. wrong address specification).

Therefore, we can perceive the trend of cyber incident, including DoS attack, by analyzing those arriving in the darknets. CAIDA publishes the darknet dataset they observed[12], and it has been used to perceiving the trend of cyber incident.

2) DDoS mitigation on wide area networks: The traffic of large scale DDoS attacks may be up to hundreds Gbps. So it is not reasonable that a service provider prepares for such large traffic.

Therefore, measures on upstream wide-bandwidth networks are discussed and enforced.

Akamai the CDN provider provides DDoS mitigation solutions[13]. It can detect and fend off DDoS attack on their upstream networks before DoS attack traffics reach the customer origin.

B. Measures against Slow HTTP DoS

Slow HTTP DoS attack will be established when the connection pool of the server is depleted. Therefore, the key point of defense against Slow HTTP DoS attack is to avoid keeping the undesirable connections.

We will introduce some known effective methods in this section.

1) Increasing the limit of simultaneous requests: It is possible to prevent the depletion of the connection pool and increase the resistance against Slow HTTP DoS attack by increasing the number of simultaneous processing of requests[14]. In apache 2.4 with Event Multi-Processing Module (Event MPM), it is configured by the `MaxRequestWorkers` directive[6]. The default value is 150.

However, it is restricted by the server performance and it is not possible to increase endlessly. In contrast, the attacker can hold many connections with low cost. Therefore the protective effect of this method is limited.

2) Shortening the timeout value: It will be more difficult for attacker to keep the connection for a long time if the timeout value is made shorter. In apache 2.4, it is configured by the `TimeOut` directive, that the default value is 60 seconds[6].

Park et al., reported the relationship between the resistance against Slow Read DoS attack and the timeout value[4]. In the experiment, they generated 500 Slow Read DoS attack connections (up to 50 connections per second) by slowhttptest.

We cite the experimental result in Figs. 1 and 2. MC/SL means `MaxCleints/ServerLimit`, that is the maximum simultaneous processing of requests in Apache 2.2 with Prefork MPM used in the experiment. During the number of child processes is reached MC/SL, Apache cannot accept a new request, that means the attack is established.

The service was stopped in 93 seconds when the timeout is 100 seconds (shown in Fig. 1). Meanwhile, the out of service time was shorten to 17 seconds when the timeout is 10 seconds (shown in Fig. 2).

However, Park et al. say in the report that too short timeout makes the Quality of Service (QoS) degraded, so it is impossible to protect from Slow HTTP DoS attack effectively by just shortening the timeout.

3) Limiting the number of requests for each client: It is possible to prevent the connection pool depleting by limiting the number of requests for each client[4]. If the limit is lower than the maximum simultaneous requests of the server, the attack by just one attacker will fail.

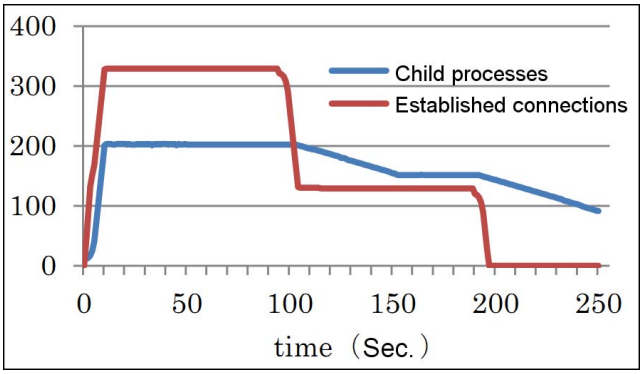

Fig. 1. Experimental result by Park et al. (Timeout 100s, MC/SL 200) (Translated into English by the quoter)[15]

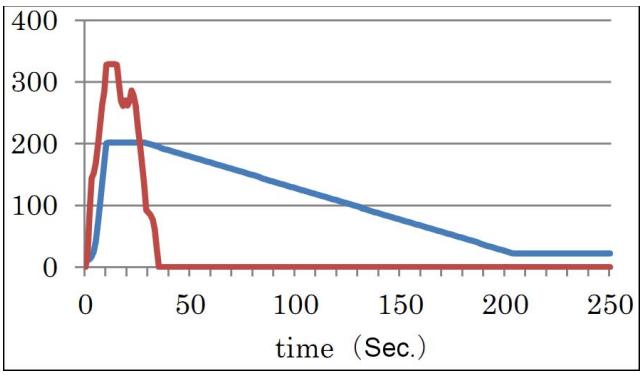

Fig. 2. Experimental result by Park et al. (Timeout 10s, MC/SL 200) (Translated into English by the quoter)[15]

We can identify the clients by the IP addresses, because the attacker must establish the TCP connection when providing Slow HTTP DoS attack. Therefore, we do not have to mind IP spoofing.

Park et al. reported the resistance against Slow Read Dos attack by limiting the number of requests at the same time for each IP address.

We cite the result of experiment in Fig. 3. In this experiment, they limited the number of requests for each IP address to 100, and attacked by one host. The result shows that the number of child processes did not hit the MC/SL (300) and the attack failed.

On the other hand, the attack succeeded during 18 seconds in another experiment for the case using 3 attackers (Fig. 4). It is inferred that the time attack established will be more longer if more attackers participated.

Usually many attackers participate DDoS attacks typically provided by a botnet. Therefore, the method limiting the number of connections for each IP address is not effective against distributed Slow HTTP DoS attack and it is a real threat.

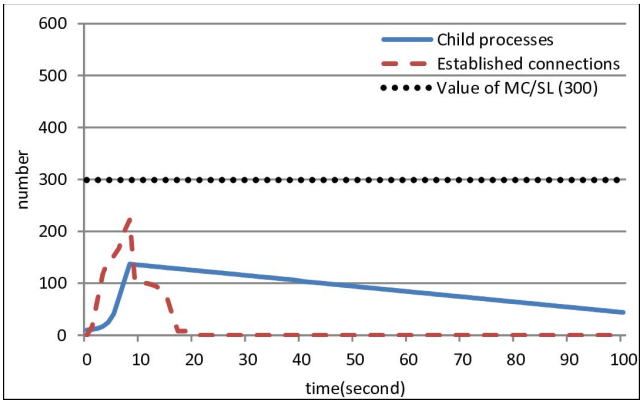

Fig. 3. Experimental result by Park et al. (Attacked by 1 host)[4]

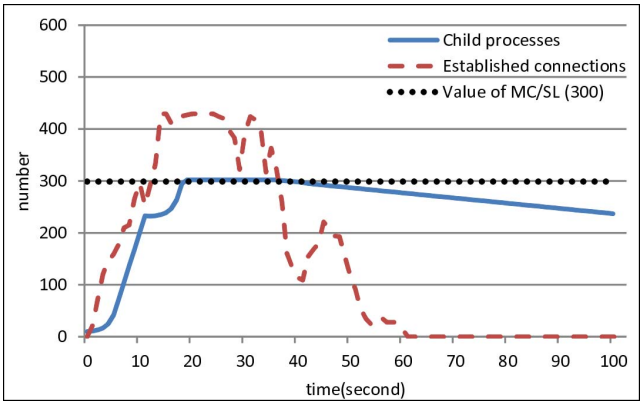

Fig. 4. Experimental result by Park et al. (Attacked by 3 hosts)[4]

TABLE I
PARAMETERS

Symbol	Definition
n	The number of processing requests.
m	The maximum number of simultaneous requests.
u	The threshold of request is tightness.
l	The threshold of process exit.
t	The threshold of duration time.

IV. Proposed Method

We propose a defence method against large-scale distributed Slow HTTP DoS attack, such as an attack provided by a botnet, in this section. This method enables effective protection by disconnecting the attack connections selectively by focusing on the number of connections for each IP address and the duration time. The details are shown below. The definition of parameters used in the explanation is shown in Table I.

This method consists of mainly three steps. Monitor the number of connections on Step 1 in a normal time. When it decided that the attack is in progress based on the number of connections, go to Step 2 and 3.

Step 1: If $n > u$, decide that the Slow HTTP DoS attack is in progress, and go to Step 2.

TABLE II
THE PARAMETERS USED IN THE EXPERIMENT

	parameter	value
Server	MPM	Event MPM
	Timeout	60s
	MaxRequestWorkers	300
	Size of content	1MiB
Attacker	Test duration	180s
	Total number of connections	500
	Connection rate	50 connections/s
	Window size (Used in only Slow Read DoS)	8-16
	Bytes to read from the receive buffer (Used in only Slow Read DoS)	5 bytes

Step 2: Extract the most frequent source IP address as the connections that continued greater than or equal to t, and disconnect all the connections from the IP address. But if there are multiple IP addresses that have same number of connections, disconnect all connections from the IP address that have greatest number of connections regardless of the duration time.

Step 3: If $n < l$, quit disconnection process and go back to Step 1. If not, go back to Step 2 and continue the disconnection process.

This method only monitors the number of connection in a normal time, therefore it does not affect the QoS.

During attack is in progress, innocent clients that finish the connection earlier than t will not be disconnected by Step 2, so it also does not affect the QoS. Even when innocent clients hold the connection for a long time because of some reason, such as slow network, or CGI scripts that take much time, it will not be disconnected because innocent clients usually do not hold such slow connections in large amounts.

When an attacker will try to attack against a server applying this method, the attacker have to finish the connection earlier than t, or reduce the number of connections for each source host to prevent the connections being disconnected. However if the attacker did so, the attacker will have to prepare many source hosts to make the attack succeed. Therefore, this method will raise the cost of Slow HTTP DoS attack.

V. EVALUATION

We will evaluate the resistance of proposed method against Slow HTTP DoS attack by experiment in this section.

A. Environment of Experimentation

In our experiment, we ran the Apache HTTP Server and our proposed method on the server. Then we ran slowhttptest on 3 computers as the attackers on the same network of the server.

We compared the resistance against three Slow HTTP DoS attack methods under two cases, without defence measure and with our proposed method.

We launch an attack from multiple attackers (i.e. distributed Slow HTTP DoS attack) by assigning 10 IP addresses for each attacker. We produced only attack traffics in this experiment. Therefore no innocent one exists.

B. The result of experiments with one attacker

We set the parameters of the proposed method to $t = 1,000,000\mu s$, $l = u = 250$ in this experiment.

The transition of number of established connections is shown in Figs.5–7. When the number of established connections is reached to `MaxRequestWorkers`, the server cannot accept a new request, meanwhile, it fallen in denial-of-service state.

In the experiment without proposed method, the number of established connections became up to 428 and the service got stopped. This number is sum of the number of processing requests (300) and the connections that are established but their requests are not accepted by Apache yet (128). The latter number is restricted by kernel configuration that is 128 in default.

On the other hand, in the experiment with proposed method, the attack connections was disconnected when the number of connections reached to l (=250) in the three attack methods. Accordingly, the service did not get stopped.

From the results, we found that the proposed method has a resistance against Slow HTTP DoS attack from just one attacker.

C. The result of experiments with multiple attackers

Next we will show the result of experiment that realized a distributed slow HTTP DoS attack using 30 attacker hosts. As the same as the experiment with one attacker, we set the parameters to $t = 1,000,000\mu s$, $l = u = 250$.

The results are shown in Figs.8–10.

In Slow HTTP Headers attack (Fig.8) and Slow Read DoS attack (Fig.10), the service got denial of service state when not applying proposed method. However, the number of connection was limited up to 250 when applying proposed method. This is a similar result to the experiment with one attacker.

However, in Slow HTTP POST (Fig.9), the service got denial service state even the proposed method applied. This result shows that the attack connections came more frequently than disconnecting.

Therefore we experimented again modifying the thresholds l and u. The proposed method starts disconnecting with fewer connections by lowering u. Further the proposed method tends to disconnect the connections from multiple clients in once disconnecting step by lowering l. We can expect to improve the resistance against Slow HTTP DoS attack by setting these thresholds properly.

Fig.12 shows the result when l was decreased to 200, without changing u. The number of connections was suppressed up to 286, and the service did not fall denial of service state.

Meanwhile, Fig.11 is the result when the thresholds was set to $l = u = 200$. The number of connections was suppressed fewer than the case with $l = 200$, $u = 250$. In this experiment, the service did not fall denial of service state neither.

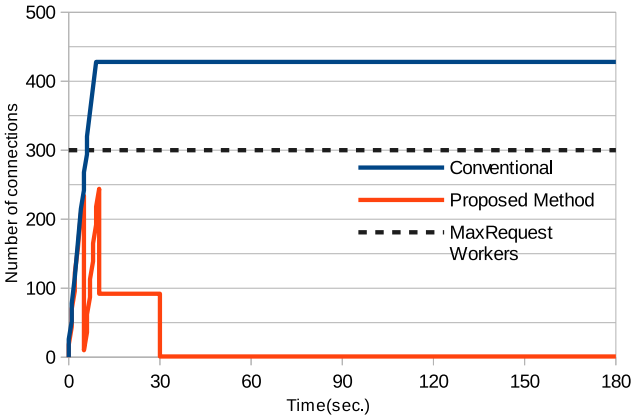

Fig. 5. Slow HTTP Headers by 1 host

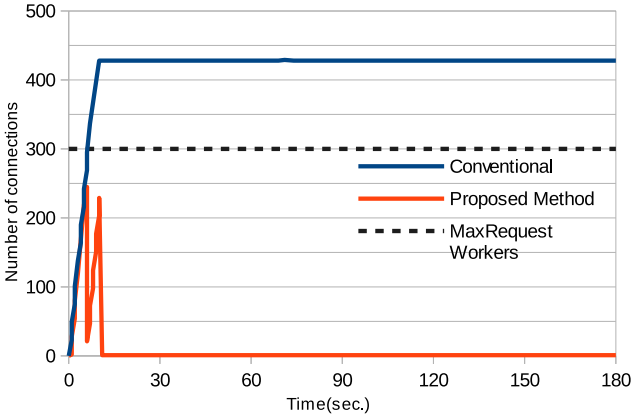

Fig. 6. Slow HTTP POST by 1 host

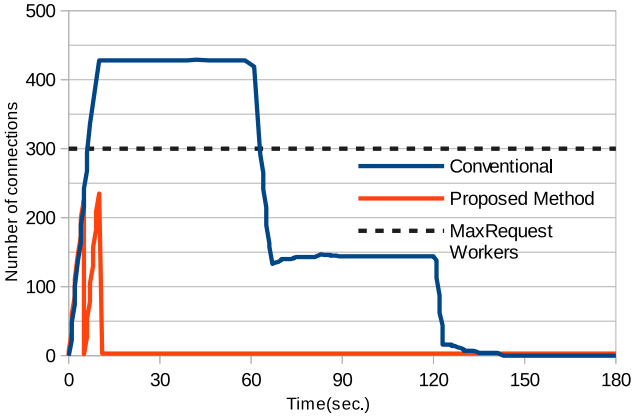

Fig. 7. Slow Read DoS by 1 host

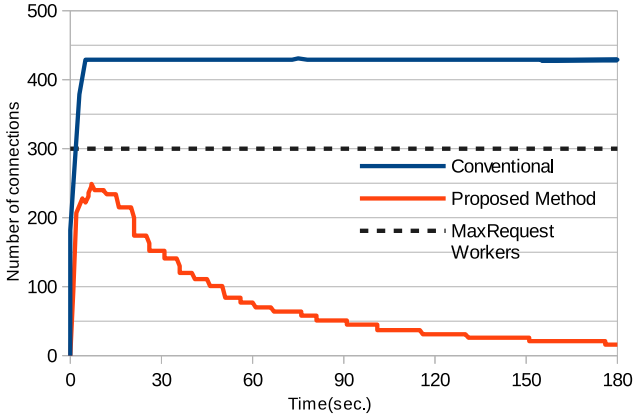

Fig. 8. Slow HTTP Headers by 30 hosts

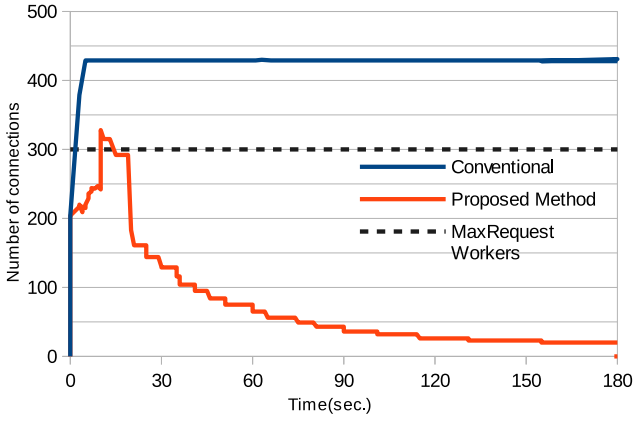

Fig. 9. Slow HTTP POST by 30 hosts

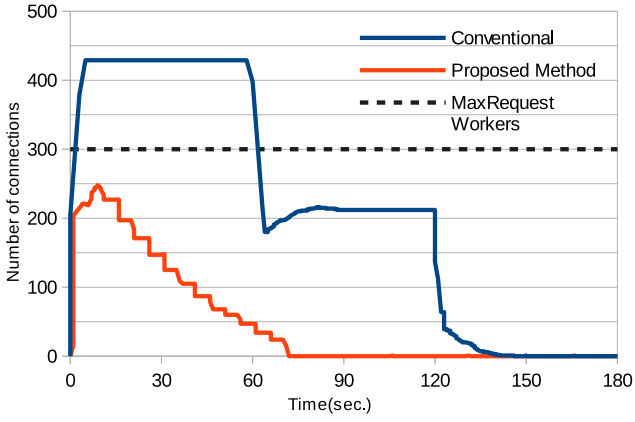

Fig. 10. Slow Read DoS by 30 hosts

However, decreasing u means that it begins disconnecting process with fewer connections. It may cause false detection and disconnecting the innocent clients. Therefore too small u will have a undesirable influence for the QoS.

VI. CONCLUSION

We proposed a defence method against Slow HTTP DoS attack and we evaluated its resistance in this paper.

As a result, we found that the proposed method had a enough resistance against a Slow HTTP DoS attack from one attacker. We also found that the proposed method can defend effectively against distributed Slow HTTP DoS attack from 30 attackers by setting the thresholds properly.

We will evaluate the resistance of this method against a large-scale distributed DoS attack in the future. We will also evaluate the resistance using different thresholds, especially t, that is common in all experiments in this paper.

In addition, we should evaluate the side effect of the proposed method to QoS because DoS attack is an attack that interferes services. We will evaluate side effect of this method to throughput and possibility of disconnecting normal clients. Evaluation of QoS will be carried out in both conditions with or without the server suffering Slow HTTP DoS attack.

VII. ACKNOWLEDGMENT

This work was partially supported by JSPS KAKENHI Grant Number 26330159.

REFERENCES

[1] Akamai Technologies. "akamai's [state of the internet] / security q3 2015 report". https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/2015-q3-cloud-security-report.pdf. Accessed: 2016-04-18.

[2] stressboss.net. "stress boss - best ip stresser". https://stressboss.net/. Accessed: 2016-04-17.

[3] BNP Media. "defending the network from real iot threats". http://www.securitymagazine.com/articles/86545-defending-the-network-from-real-iot-threats. Accessed: 2016-04-17.

[4] Junhan Park, Keisuke Iwai, Hidema Tanaka, and Takakazu Kurokawa. "analysis of slow read dos attack and countermeasures". In *Computer Security Symposium 2014*, pages 354–361, October 2014. 2B1-4.

[5] Akamai Technologies. "slow dos on the rise". https://blogs.akamai.com/2013/09/slow-dos-on-the-rise.html. Accessed: 2016-04-22.

[6] The Apache Software Foundation. "core - apache http server version 2.4". http://httpd.apache.org/docs/2.4/en/mod/core.html. Accessed: 2016-04-22.

[7] Hacking News. "anonymous released an isis hacking guide". http://www.hackingnews.com/security/anonymous-released-an-isis-hacking-guide/. Accessed: 2016-04-22.

[8] GitHub. "home shekyan/slowhttptest wiki". https://github.com/shekyan/slowhttptest/wiki. Accessed: 2016-04-22.

[9] Qualys. "are you ready for slow reading?". https://blog.qualys.com/securitylabs/2012/01/05/slow-read. Accessed: 2016-04-22.

[10] SlashdotMedia. "average web page approaches 1mb". https://tech.slashdot.org/story/11/12/22/2015231/average-web-page-approaches-1mb. Accessed: 2016-04-22.

[11] Team Cymru. "the darknet project". http://www.team-cymru.org/darknet.html. Accessed: 2016-04-23.

[12] Center for Applied Internet Data Analysis. "the ucsd network telescope". http://www.caida.org/projects/network_telescope/. Accessed: 2016-04-23.

[13] Akamai Technologies. "ddos mitigation". https://www.akamai.com/us/en/resources/ddos-mitigation.jsp. Accessed: 2016-04-23.

[14] Qualys. "how to protect against slow http attacks". https://blog.qualys.com/securitylabs/2011/11/02/how-to-protect-against-slow-http-attacks. Accessed: 2016-04-25.

[15] Junhan Park, Keisuke Iwai, Hidema Tanaka, and Takakazu Kurokawa. "the research of slow read dos attacks to web server". In *SCIS 2014, The 31st Symposium on Cryptography and Information Security*, pages 1–7, January 2014. 2C1-4.

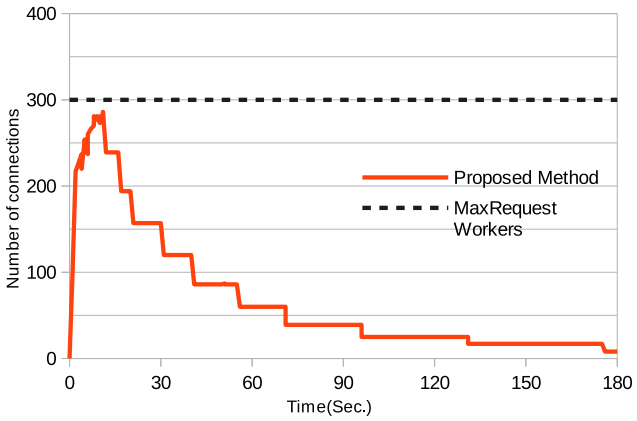

Fig. 11. Slow HTTP POST by 30 hosts ($l = 200$,$u = 250$)

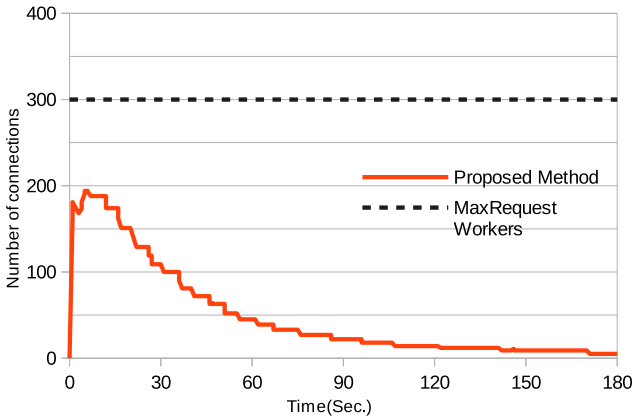

Fig. 12. Slow Read DoS by 30 hosts ($l = 200$,$u = 200$)