Future Computing and Informatics Journal

Volume 2
Issue 2 (2020) *2017, Volume 2, Issue 2*

Article 3

2017

A new method to reduce the effects of HTTP-Get Flood attack

Hamid Mirvaziri
Computer Department, Engineering Faculty, Shahid Bahonar University of Kerman, Iran,
hmirvaziri@uk.ac.ir

Available online at www.sciencedirect.com

ScienceDirect

Future Computing and Informatics Journal 2 (2017) 87–93
http://www.journals.elsevier.com/future-computing-and-informatics-journal/

A new method to reduce the effects of HTTP-Get Flood attack

Hamid Mirvaziri

Computer Department, Engineering Faculty, Shahid Bahonar University of Kerman, Iran

Received 21 August 2016; revised 5 May 2017; accepted 2 July 2017
Available online 29 July 2017

Abstract

HTTP Get Flood attack is known as the most common DDOS attack on the application layer with a frequency of 21 percent in all attacks. Since a huge amount of requests is sent to the Web Server for receiving pages and also the volume of responses issued by the server is much more than the volume received by zombies in this kind of attack, hence it could be done by small botnets; in the other hand, because every zombie attempts to issue the request by the use of its real address, carries out all stages of the three-stage handshakes, and the context of the requests is fully consistent with the HTTP protocol, the techniques of fake address detection and anomaly detection in text could not be employed. The mechanisms that are used to deal with this attack not only have much processing overload but also may cause two kinds of "False Negative" (To realize wrongly the fake traffic as the real traffic) and "False Positive" (To realize wrongly the real traffic as the fake traffic) errors. Therefore a method is proposed that is able to adapt itself to the traffic by the use of low processing overload and it has less error than the similar systems and using this way.

Keywords: DDOS; HTTP-GET flood attacks; IOSEC

1. Introduction

1.1. Methods of HTTP-GET flood detection

Various methods have been suggested for detection and intensity reduction of HTTP Get Flood attack by different people, some of which are listed in the following.

- Using the puzzle CAPTCHA [1];
- Detecting the same sequence of requests [2];
- Detecting the speed of browsing pages [2];
- Predicting the attack by the use of attack rate [3];
- Detecting the relationship between users' requests [4];
- The number of requests in the specific period of time [5,17];
- Scoring to the connection [6];
- Detecting the pattern of requests [7];
- Determining the extent of the existent risk in accessing to the pages [8];
- Using the infrastructure of the detection complex of DDOS (IDDI)[1] attacks [17];
- Detecting the abnormality in the requests [9];
- Using several methods sequentially [10];
- Load balancing [11,12,15];
- Using priority queues with different bandwidth [12];
- IP Hopping—Moving target defense mechanism [13];
- Using hidden cache of the browser [14];
- Blocking by use of Access Control Lists (ACL) [14];
- Using hardware methods [16].

Among the above methods, "The number of requests in the specific period of time" method has a small processing load and is one of the most practical ways that could reduce the intensity of this kind of attack. In this method, determining the

E-mail address: hmirvaziri@uk.ac.ir.
Peer review under responsibility of the Faculty of Computers and Information, Future University in Egypt

[1] Integrated DDOS Defense Infrastructure.

http://dx.doi.org/10.1016/j.fcij.2017.07.003

number of requests in a pre-determined period of time is taken into account, and if it exceeds from a specific threshold, then it is identified as traffic attack and consequently blocked. In this method, the allowed number of each user's requests is significantly important as well as the determining of time interval, so that the extent of accuracy and correct performance of this method depends completely on these two parameters. For example, even so the number of 100 requests in 10 s is mathematically equal to 10 requests in 1 s, but each of them shows a different performance when it is required to have legal selection in order to distinguish the real traffic from the fake one. The system IOSEC uses this method, and in this paper, it is used as a criterion to measure the extent of accuracy in the proposed method.

In IOSEC system which is installed as a plug-in on the context management system Word Press, a request from every user is identified in every 500 ms which is allowable by default, and if the number of requests sending by a user exceeds from the mentioned number in this period of time, a page like Fig. 5 is displayed to the user. This page is changed by default after 10 s and the user requested page is displayed. If the user resends a request before finishing the 10 s then the mentioned page is shown again for him and the inside timer will start recounting from 10. In this system, the mentioned page can be sent unlimitedly to the user, unless the user resolves the puzzle at the bottom of the page. The user by resolving the puzzle that is located at the bottom of the page introduces him as the legal user, and he will not see such a page anymore (after he is added to the white list). It is trivial that if a hacker can success to solve the puzzle for each agent and put address of that agent in the white list, this system would not resist against the issued attack from that agent anymore.

2. Materials and methods

2.1. Discussion about the proposed method

Most of introduced methods dealing with HTTP Get Flood attack are depend on the analysis of the site's traffic at the non-attack times; and due to using different parameters, they have processing and storing overload and do not have much functionality in the practical environments. Among the mentioned methods, “the determination of the number of allowed requests in a specific time” has a small overload and is usable in the practical environment, but the determination of the mentioned parameters in this method requires gathering the traffic history for each site; therefore, this method is not easily movable, and also since traffic of a site in different times is influenced by the emotions of users during their visit, so this method does not have high accuracy. Simplicity and applicability of this method caused that a technique is created which has high accuracy in addition to have the ability of implementing by hardware and software. It has also high portability without any knowledge of content and traffic history of the site. In this paper, HGFMS method is introduced with portability and traffic adaptability as well as hardware applicability. The location of this system is shown in Fig. 1.

HGFMS makes the number of requests automatic and time interval in different states is consistent with traffic by introducing a new criterion as server status and by using of random time delay mechanisms as well as by using of trap link. Therefore it can be used without having a traffic history to display more accurate performance.

In HGFMS, we assume the maximum response time of the server to the requested pages (not components within the page) is in second unit in the main bottleneck and also the main user uses at most 40 percent of the server power in the above-mentioned criterion so the current state of the server can be divided into normal and abnormal. In other words, if server is working with more than 40% of its power, the abnormal state is considered; the traffic of users is suspected and a page is shown to the user as shown in Fig. 2. In the mentioned page, there is a button to continue which is activated after the displayed time in the button and by pressing it the user is directed to the requested page.

To deal with intelligent agents, the following cases are considered in the waiting page shown to the user:

1. The length of delay time is selected randomly in the interval of 30–60 s, and the user is allowed to press the mentioned button after finishing the mentioned time, therefore if the time distance between the requests of participant agents in the attack is more than the specified amount or less than it, the traffic attack would be detected.
2. The link of requested page is not placed directly in backside of the button; thus, if a request is sent to the link in the backside of the button before finishing the appropriate time in the page, this indicates that request is issued by an intelligent agent.
3. Considering that in the worst situations, intelligent agents act exactly like downloader programs of the site and attempt to issue a request to the all built-in links within the page, the trap link has been used at the beginning of all pages in the site. If there is a request to the trap link, it immediately put the related IP into the black list and blocks other requests from that IP.

It should be noted that intelligent agents usually use multi threads at the same time to issue a request to the links within the page and therefore the number of available trap links within the page and even the location of them is effective in the number of responses the agent receives before being blocked; for example site downloader software and intelligent agents with similar behavior can be blocked by placing the trap link at the beginning and at the end of the first page of those kinds of sites that have only a few limited links to select language.

In the flowchart of proposed system which is illustrated in Fig. 3 the following structures have been used:

1. List of requests to access to the page: all requests to receive the pages together with the receiving time and the

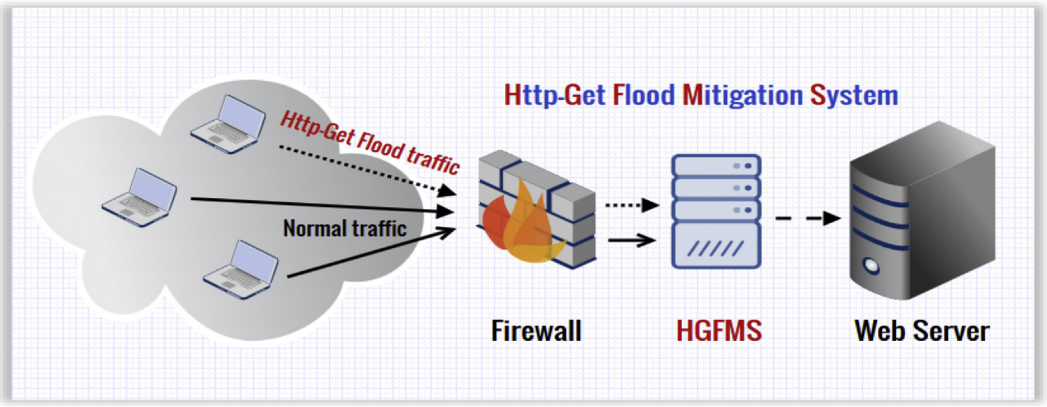

Fig. 1. The location of the proposed system.

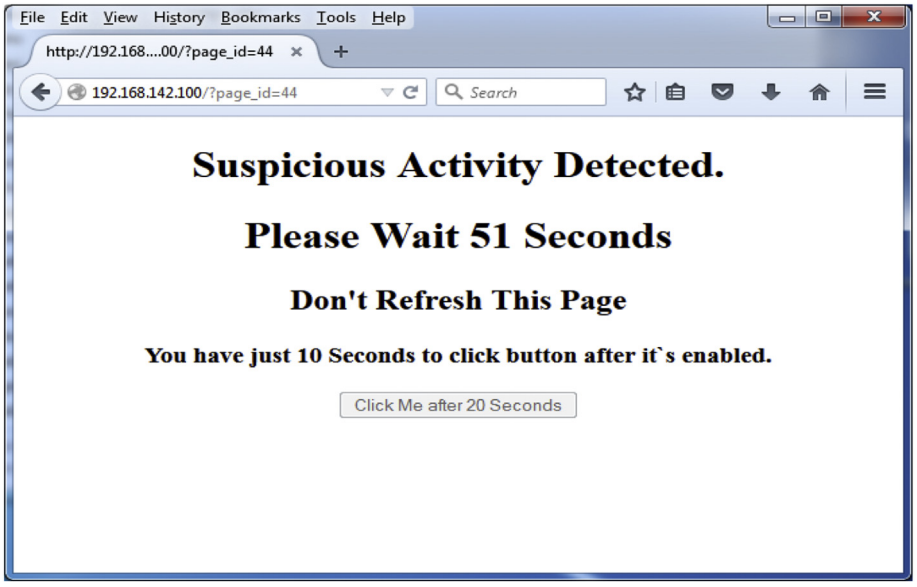

Fig. 2. Waiting page to confirm the identity of the requester.

IP address of the requester are stored in a table and by the use of this table the number of requests of every IP at is calculated a time unit.

2. The table of suspicious IPs: those IPs that are suspicious and send them waiting page, are kept in this list. Also the address of requested page as well as a list of suspecting times is kept in this list. In the proposed system, each IP could be suspected three times at most; in the other words, the time-delay page would be shown at most three times for each IP address and if the related button is not pressed during these times, that IP address would be moved to the black list, and its future communication would be blocked.
3. The history list: the suspicious IP addresses and times that managed to pass the time-delay stage successfully when the server was in normal state, are placed in this list. This list will be used to enhance the threshold amount of the number of requests in the specified time interval.
4. The black list: the IP addresses that are known as unallowable and are not allowed to use the server are kept in this list.

2.2. Theory of the proposed system

In the beginning, the request sender's address and the addresses within black list are investigated, and only the addresses that are not within the black list is allowed to continue; then the page request is investigated and if a request for the forbidden page is received, its sender address is placed in the black list and their specifications are removed from the history and specious list as shown in Fig. 3.

In the next stage, the speciousness of the sender's address is investigated, and if it is not in the specious list, there are two cases:

1. The number of user requests is more than the threshold at the time unit. In this case, the user is suspected and a time-wait page will be sent to him. The user who receives the time-wait page is allowed to be reentered to the site only by a click on the button inside it. This button, is deactivated for 30–60 s in the beginning and then it is activated so the user should press the button at most 10 s after

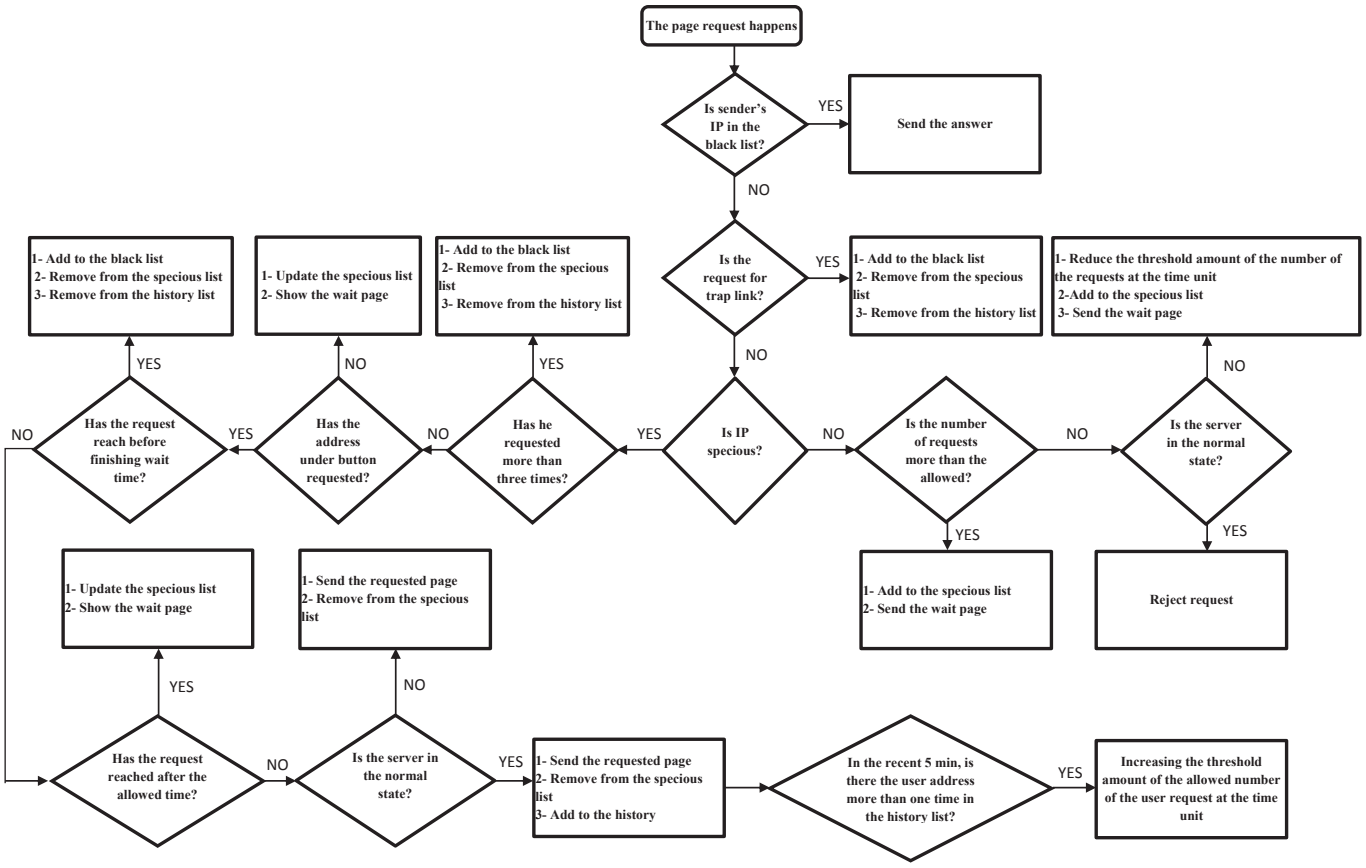

Fig. 3. Flowchart of the proposed system.

activating. If, for any reason, the user could not do this, or he recalls the page or presses the back button of his browser, this page is shown to him again. In total, if the page is seen three times by the user, but the user does not press the button inside it, then that IP address will be blocked at the next visit. On the other way, if the suspicious user presses the related button at the specified time and the server was in the normal state, in addition to sending the considered page to the user and removing his address from the list of suspicious users, the address and the time of accepting his request is recorded in the history list. Then, it is investigated at the defined time interval if the above user has repeated the above stages more than once or not. If the user address has been repeated more than once at the specified length of time, it means that, probably, the defined threshold for the number of requests at the time unit is so small that result in this event, so one unit is added to it. Thus the system detects and applies the optimal value by increasing and reducing the mentioned threshold at different times. It should be noted that if the user presses the button at the specified time but the server is not in the normal state, then a request page will be shown to him and his address will not be added to the history list; therefore the history list includes only the accepting times and the addresses of the specious users who have stepped forward the waiting stage correctly when the server was in the normal state.

2. The number of user requests is less than the threshold at the time unit. In this case, the server status should also be investigated and the appropriate behavior should be done with respect to its status. If the server is in the normal state, then the requested page is sent to the user but if the server is not in the normal case, there are another two cases:
   a. The defined threshold for the number of allowed requests is more than the needed value at the time unit.
   b. The server has been attacked by a large botnet.

To prevent the first possibility, defined threshold is subtracted by one if it is higher than one and in the second case, send a time-wait page to all the users (all of them are suspected) in order to understand the human nature of the requests issuers (see Table 1).

2.3. Implementation of HGFMS (proposed method): practical approach

To implement HGFMS, the content management system Word Press was used to create a site with seven designed pages

Table 1
Specifications of the server system.

Operating System	Windows 7 Ultimate x64	
CPU	AMD Athlon 64 X2 5000	
RAM	4 GB	
Hard Disk Capacity	320 GB	
Speed of the Network Card	100 Mb/s	
Launched Version of the Server	XAMPP v5.6.11	PHP 5.5.27 MySQL 5.6.25 Apache 2.4.12

Table 2
Specifications of the pages of the test site.

The address of each page	The number of the requests for receiving the page and its internal components
http://192.168.142.100/	15
http://192.168.142.100/?page_id=74	25
http://192.168.142.100/?page_id=96	26
http://192.168.142.100/?page_id=44	18
http://192.168.142.100/?page_id=23	13
http://192.168.142.100/?page_id=112	12
http://192.168.142.100/?page_id=12	14

according to Table 2 and it is installed on a system with the specifications shown in this Table. Both systems; HGFMS and IOSEC; are installed as a plug-in on the mentioned content management system and reaction of them is recorded by activating each of the mentioned plug-ins separately in every stage and the attack is performed.

By the use of mentioned content management system and the addition of the following line in the header file, the trap link was added to the beginning of all pages. As it is clear from the html code, to hide the link from the users' eyes, no text has been written for it. It should be noted that the trap link is hidden from the users' eyes, but it is visible for the site download software, such as Teleport, Pro and intelligent agents that operate within the same way, and they should send a request to receive them like other links inside the page,

<ahref=“http://192.168.142.100/?page_id=BlockMe”>

Employed topology to test HGFMS system is shown in Fig. 4. In this implementation, the operating system is Linux.BoNeSi and LOIC softwares are used to simulate the traffic of the HTTP Get Flood attack and F5[2] attacks respectively. In order to convergence of this system and to obtain an appropriate value for threshold of allowed number of the request, a client attempted to send his requests, and on average, after 5 h and passing 4 times of waiting page, the threshold value was changed from its initial value to 5. The

[2] In attacks F5, the browser, under the effect of the available script inside the site or through holding the button F5 by user (the Refresh operations in many browsers is done by the button F5), attempts to send sequential and same requests to the regarded site.

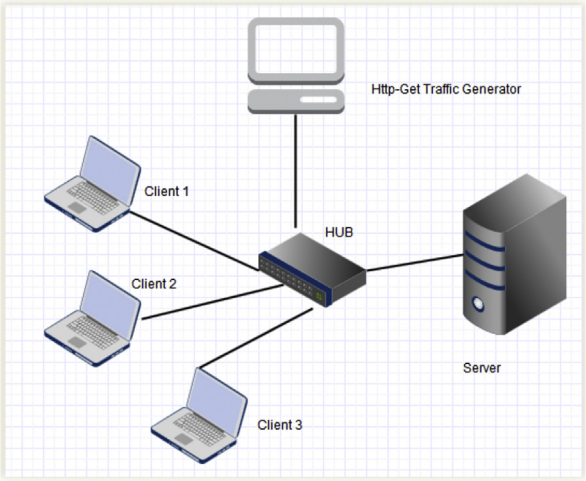

Fig. 4. Topology used in the implemented environment.

Table 3
Operation of the proposed system and measured time by a typical user.

Operation	Time
Initial value of allowed number of the requests in 30 s	1
Duration of adapting to the traffic	5 h
Average number of requests	1133
The number of times seeing waiting page	4
Final value of the allowed number of requests in 30 s	5

Table 4
Values in the implementation of the proposed method.

Title	Value
The maximum number of the requests in second	15
Threshold of the normal server performance	6
Duration to calculate the allowed number of the user requests	30 s
Initial value for the allowed number of the user requests	1
Time interval between the requests of legal users	Randomly between 2 and 30 s
Duration of the wait shown on the waiting page	Randomly between 30 and 60 s
Allowed duration to press the available button on the waiting page	10 s
Time of holding and applying the history	5 s

information related to this period is presented in Table 3. Also, the values that are used to implement the proposed system are given in Table 4.

3. Comparison between proposed and similar systems

To compare the performance of HGFMS with similar systems, IOSEC HTTP Anti Flood/DoS Security Gateway Module, was chosen [5]. Exploiting the mechanism of a number of

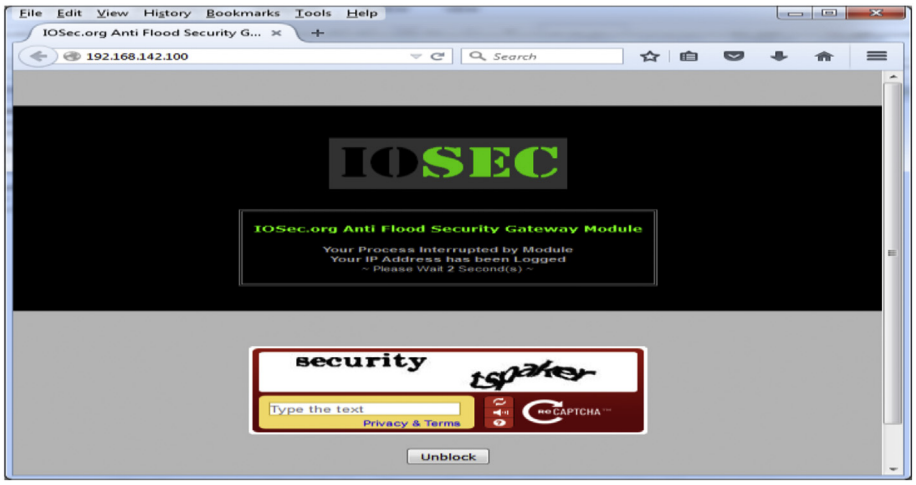

Fig. 5. Time delay page shown by the IOSEC plug-in.

requests at the time unit is used to detect the traffic of attack. This system is available in the form of a plug-in on the content management system. In this plug-in, the allowed number of user's requests is calculated in the time unit, and if it exceeds from the pre-determined specific value, a page is shown to the user similar to the Fig. 5. The settings and the specifications of both systems are given in Table 5.

4. Results

In this article two methods, IOSEC and HGFMS (proposed method) is implemented and after carrying out different attacks on both mentioned methods the results in Table 6 are achieved. According to these results, HGFMS behave much more effective than IOSEC dealing with a variety attack

Table 5
Comparison of the proposed method with the method introduced in Ref. [5].

	IOSEC	HGFMS
The time interval of the calculation of the requests	500 ms	30 s
The number of requests in the time interval	2	Variable
The benchmark to calculate the server capacity	Does not have	Request for page in seconds
The way to seal with the suspicious traffic	Sends the time waiting page together with the puzzle CAPTCHA	Sends the wait page
The number of successive suspiciousness to an address	Infinite	At most three times
Permanent removing of the suspicious traffic	Dose not have	By the use of the black list
Adaptability to traffic	Does not have	Has
The calculation benchmark	Number of all requests at time unit	Number of the requests to receive the page at time unit
Criterion of attack detection	Exceeds from the defined threshold for the maximum of user requests at time unit	Exceeds from the defined threshold for the maximum of user requests for the user page at time unit
Ability to deal with the intelligent agents	Does not have	Has (Using the trap link)
The ability to deal with the agents that have the capability of adjusting the request rate	Does not have	Has
The possibility to introduce the agent as legal user	using the resolution of the puzzle CAPTCHA and recording the agent's address in the white list	Does not have

Table 6
Comparing the performance of the system dealing with the attack of the intelligent agents.

	IOSEC	HGFMS
The number of the received files with 10 threads at the same time	44 Success rate: 29%	27 Success rate: 56%
The number of the received files with 1 thread in an interval of 1 s	62 Success rate: 0%	29 Success rate: 53%
The number of the received files with 1 thread in an interval of 30 s	62 Success rate: 0%	29 Success rate: 53%
The extent of responsiveness to users at time of the attack F5 by the use of one agent	0%	100%
False positive	12%	1%

factors and provides conditions for the server to serve its users even at the time of attack.

5. Conclusion and suggestion

By taking advantage of new features of HGFMS, it is concluded that it has a better performance comparing with similar system. The feature of trap link is caused that the proposed system can block the traffic of the downloader software of the site and the intelligent agents that have similar behavior. Also, it has been able to adapt the allowed number of requests to the traffic and situation of the server by use of a new parameter called the server status without the need to have the site's traffic history. Therefore it has the ability to deal with different rates of attacks and will have fewer errors than the similar methods. Moreover, using waiting page with variable time instead of CAPTCHA puzzle makes it user-friendly and natural secure system. It has high portability because of independency on site content. Hardware implementation is suggested in order to separate the processing load related to this system. Also, it is recommended that Tarpitting is used instead of removing requests of the users who are put in the black list to reduce processing load of the server and impose the processing load on the agent which leads them to be removed.

References

[1] Ko N, Noh S, Park JD, Lee SS, Park HS. An efficient anti-DDoS mechanism using flow-based forwarding technology. 9th international conference on optical Internet (COIN), Jeju, July, 1–3. 2010.
[2] Yatagai T, Isohara T, Sasase I. Detection of HTTP-GET flood attack based on analysis of page access behavior. In: Proceedings of IEEE Pacific Rim Conference on communications, computers and signal processing, Victoria, Auguest; 2007. p. 232–5.
[3] Thapngam T, Yu S, Zhou W, Beliakov G. Discriminating DDoS attack traffic from Flash Crowd through Packet Arrival patterns. IEEE conference on computer communications workshops (INFOCOM WKSHPS), Shanghai, April. 2011. p. 952–7.
[4] Xie Y, Yu S. A large-scale hidden semi-Markov model for anomaly detection on user browsing behaviors. IEEE/ACM Trans Netw 2008;17(1):54–65.
[5] Muharremoglu G. Web application level approach against the HTTP Flood attacks IOSEC HTTP Anti Flood/DoS Security Gateway Module. 2012. Available from: http://www.iosec.org [accessed 10.07.14].
[6] Beitollahi H, Deconinck G. Tackling application-layer DDoS attack. The 3rd international conference on ambient systems, networks and technologies, Canada, Aug. 2012. p. 432–41.
[7] Lu WZ, Yu Sh. An HTTP flooding detection method based on browser behavior. International conference on computational intelligence and security, Guangzhou, November. 2006. p. 1151–4.
[8] Lin CH, Lee ChY, Liu JCh, Chen ChR, Huang ShY. A detection scheme for flooding attack on application layer based on semantic concept. International Computer Symposium (ICS), Tainan, December. 2010. p. 385–9.
[9] Cid D. Analyzing popular layer 7 application DDoS attacks. 6 2014. Available from: https://blog.sucuri.net/2014/02/layer-7-ddos-blocking-http-flood-attacks.html.
[10] Giralte LC, Conde C, Diego IM, Cabello E. Detecting denial of service by modelling web-server behavior. Madrid, Spain: Computer Architecture and Technology Department, Universidad Rey Juan Carlos; October 2013. p. 2252–62.
[11] Eddy W. TCP SYN Flooding attacks and common mitigations: IETF. RFC 4987. 2007.
[12] Chee WO, Brennan T. Http post. OWASP AppSec DC, Canada, Nov, 15–81. 2010.
[13] Yang-Seo C, Jin-Tae O, Jong-Soo J, Jae-Cheol R. Integrated DDoS attack defense infrastructure for effective attack prevention. 2nd international conference on Information Technology Convergence and Services (ITCS), Cebu, August. 2010. p. 1–6.
[14] Mohamed Ibrahim AK, George L, Govind K, Selvakumar S. Threshold Based Kernel Level HTTP Filter (TBHF) for DDoS mitigation. Int J Comput Network and Inf Security (IJCNIS) 2012;4(12):31–9.
[15] Spagna S, Liebsch M, Baldessari R. Design principles of an operator-owned highly distributed content delivery network. IEEE Commun Mag April 2013;51(4):132–40.
[16] Jin J, Nodir N, Im C, Nam SY. Mitigating HTTP GET Flooding attacks through modified NetFPGA reference router. 1-st Asia NetFPGA Developers Workshop, Daejeon, Korea, June, 1–7. 2010.
[17] Choi Y, Oh J, Jang J, Ryou J. Integrated DDoS attack defense infrastructure for effective attack prevention. 2nd international conference on information Technology convergence and Services (ITCS), Cebu, Aug ,1–6. 2010.