Computer Communication & Collaboration (Vol. 4, Issue 2, 2016)
ISSN 2292-1028(Print) 2292-1036(Online)
Submitted on May 25, 2016
DOIC: 2292-1036-2016-02-003-67

Real time calibration of DDoS blocking rules for Web Servers

Sujatha Sivabalan (Correspondence author)
Electrical and Computer Engineering, RMIT University
Swanston Street, Australia
Email: s3365148@student.rmit.edu.au

Dr. P J Radcliffe
Senior Lecturer, Electrical and Computer Engineering, RMIT University
Swanston Street, Australia
Email: pjr@rmit.edu.au

Abstract: Protecting web servers from Distributed Denial of Service (DDoS) attacks in real time is a critical challenge for any security system. Several methods have been proposed to differentiate attack traffic from normal human traffic and flash traffic but the normal result is to punish both the attack traffic and at least the legitimate (and possibly profitable) flash traffic. This research has developed a novel, adaptive, real-time scoring algorithm to provide a dynamic and effective detection mechanism for a web server. A very occasional "Are You a Human" (AYAH) page is used to calibrate detections rules which are then applied to the rest of the traffic. The real-time scoring system is implemented on an Apache web server and uses shared memory to interact with a daemon to stop, slow, or allow a user request.

Keywords: Application-layer, Distributed Denial of Service (DDoS), Website

1 Introduction

The explosive growth of the Internet has changed our perception of the "normal" way of doing things. Websites such as online ticket booking, online share trading, and many online services make life easier and save money. Even many small and mid-size companies have managed to build quite profitable online businesses which in turn benefit the country's economy. The sophisticated and advanced natures of Internet technologies, as well as being a source of strength, are also a source of vulnerability. Any online business can be held to ransom by a concerted DDoS attack.

Denial of Service (DoS), aims to deny the service of network resources to legitimate users. The hardest form of DoS to block is Distributed Denial of Service (DDoS). DDoS may involve hundreds or thousands of zombie machines to overwhelm target network devices such as routers and server. This may result in short or long-term interruption to services of a legitimate host connected to the Internet [1, 2].

DDoS attacks mostly concentrated on layer 3 and layer 7 of the Open System Interconnection (OSI) architecture. The layer 3 DDoS attacks are called as Net-DDoS attacks and the layer 7 DDoS attacks are mentioned as App-DDoS [1, 2]. Blocking DDoS attacks is made even harder by a new form of user behaviour- the flash crowd. Flash traffic or a flash crowd is caused when many Internet users respond to an event at the same time, for example, when ticket sales for a concert first open, online shopping bargain days, or simply a news article or blog post. Flash traffic may induce a large increase in the number of legitimate users attempting to access a website simultaneously which stresses the network links of servers. Security devices usually interpret the flash crowd as an attack and limit the traffic to the server thus causing a financial loss to the site owners and annoyance to the users.

Detection of Net-DDoS attacks is a well-understood area and existing techniques can block attacks without losing (profitable) user traffic. The situation is very different with App-DDoS attack traffic, blocking is seldom successful because both flash traffic and attack traffic have the same characteristics; typically bursty with very high volumes. There has been some work published on blocking App-DDoS, typically generated by Zombies. All published DDoS detection mechanism [14-24] are not real time in nature and so punish flash traffic.

The main aim of this research is to mitigate DDoS attack traffic without punishing the profitable legitimate traffic with originates from a human user. The proposed architecture is novel and it is based on scoring DDoS detection rules in real time so that successful rules may be promoted and used more, and unsuccessful rules can be demoted and used less or disabled.

This paper is organized as follows: Section 2 describes existing work that attempts to block Net-DDoS and App-DDoS attack and points to why the App-DDoS methods are limited in their efficacy. Section 3 explains the novel architecture and algorithms that adaptively block App-DDoS. Section 4 describes our experimental system implementation and the results obtained. Sections 5 discuss future work. Section 6 concludes the paper and discusses the main successes and limitations of the work.

2 Existing methods

Net-DDoS attacks usually fall into one of the following categories; ICMP flooding, SYN floods and UDP flooding. Commonly a DDoS attack at the network layer is based on spoofing IP addresses [3]. This can be detected and blocked by network devices with IDS (Intrusion Detection System) capability. This activity is independent and transparent to the target server which does not need to contribute anything to this effort. Net-DDoS attacks are not that difficult to detect as stateful packet inspection can detect such activity. As a result, there are many commercial products with Net-DDoS detection and blocking capability and considerable research in the area [4-13].

App-DDoS attacks are much harder to detect than Net-DDoS attacks because the traffic looks much like a real user. In a well-crafted zombie attack, each zombie host is generating a traffic level that is not unreasonable, has a legitimate IP address, and is not doing anything with network protocols which is unusual. The survey by Durcekova et al [14] proves the necessity of new research methods on DDoS attack detection at the application layer (App-DDoS).

At the application layer, based on a statistical approach, author Ranjan et al [15] implemented rate-limiting as the primary detection mechanism and Yen et al [16] limits any source that makes random web page requests. This statistical analysis is used only for monitoring the abnormal behavior but does not differentiate between legitimate traffic and attack traffic. J. Yu et al [17] mitigate DDoS attack at the application layer based on the Trust values. Trust values are calculated using server bandwidth and the detection is done once the server exceeds the threshold. It would be better to detect an attacker before server becomes busy. The method also penalizes legitimate users along with attack traffic.

Two authors, Katkar and Zaman worked with feature selection. Katkar et al [18, 19] used naive Bayesian classifiers with numeric to binary data pre-processing for DDoS detection. Zaman et al [20] implemented a light weight IDS to overcome poor detection rate based on two different approaches. The first approach uses a feature selection approach by applying Fuzzy Enhanced Support Vector Decision Function (Fuzzy ESVDF) algorithm and the second approach uses an IDS classification scheme. The research methods [18-20] aim to select the best trained dataset and then use that data set for future detection. Smart attacks vary their features with time and so an analysis with a trained dataset will miss such attacks and may punish new legitimate traffic.

In addition to previous comments all the above methods and some other work [21-23] are not suitable for detecting DDoS attacks especially when flash traffic is involved. Kandula et al [24] and Boyd et al [25] employed CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart) to detect DDoS attacks. Every time a web page is visited, or the first time a user enters a website, the users must solve a puzzle or enter the letters shown in the image to visit the web page. This can be very annoying for a user and may well result in them leaving the website. As an example of this annoyance the author David Pogue [26] refers to CAPTCHAS as "Computer Annoying People with Time-wasting Challenges". Researchers Yi Xie et al [1, 2] monitors the App-DDoS attacks in a different way using a hidden semi-Markov model. This detection method is flawed because new legitimate flash traffic will not have the same statistics as older traffic and so will be misinterpreted as attack traffic. The Markov chain model is based on the user behavior from the first web page access and does not consider the variations in the viewing times of the web pages.

All the research listed above use methods that punish legitimate traffic as well as the attack traffic.

The detection strategies used by commercial products such as Cisco and Juniper are very interesting. Cisco Adaptive Security Appliance (ASA) [27] detects DoS attack by using stateful packet inspection (SPI) and Stateful Firewall Check. This provides an added layer of network protection but it not suitable for application level DDoS attacks because it blocks the legitimate user while blocking DDoS.

Juniper Intrusion Detection and Prevention system (IDP) [28-30] perform multistage analysis from connection monitoring to protocol analysis, to botclient classification and maintains the state for each protected server. Juniper Application-level distributed denial-of-service (DDoS) protection use a hit rate threshold parameter to classify a client from benign user to DDoS attacker. The hit rate threshold is based on the number of contexts seen in specified interval. The context is an application protocol context such as http-url and dns-cname; which amounts to a blacklist check. As mentioned earlier, if both attack traffic and flash crowd target the website, the hit rate threshold value will see the legitimate as the attack traffic and result in blocking the profitable legitimate user. The blacklist style checking will not detect new attackers or well crafted DDoS zombie-style attacks.

The App-DDoS detection mechanisms described in the literature as well as in the well-known commercial products may have some App-DDoS detection ability but they also have significant problems and so better methods must be developed. In particular, the described methods punish the legitimate traffic as well as attack traffic which can reduce the accessibility and profitability of Internet services. A key weakness is the inability to respond to changing conditions such as legitimate flash traffic.

3 Novel real-time scoring algorithm

Our novel architecture [31] provides the basic ideas for developing algorithms to protect web servers from DDoS attack. Further research has developed an algorithm based on scoring attacker detection rules in real time as shown in Figure 1. This new and effective real-time scoring algorithm will provide greater accuracy in detecting and blocking attacks in a dynamic traffic environment.

In previous algorithms found in the literature, the user's request will undergo signature analysis based on a fixed set of unchanging rules. In the proposed method, multiple blocking rules are calibrated using a very occasional CAPTCHAs or AYAH (Are You a Human) page, note the "1 in N" pathway in Figure 1. This will determine if a user is a human or an attacker though not with perfect certainty. These 1 in N user requests are also evaluated by all rules, both active and inactive. If the rule correctly predicts the user type (human or attacker) then it's "rule_accuracy" rating is incremented, if it incorrectly predicts the type of user the accuracy rating is decrement, if the rule cannot offer an opinion then it's accuracy rating is left unaffected. The rating is windowed to between zero 100. Incrementing or decrementing may not go outside this range. Any user request from this 1 in N group that goes to an AYAH page, and is determined not to be a human, will be blocked.

All user requests apart from that small number sent to an AYAH page get sent directly to the blocking algorithm in Figure 1. This applies only active rules, typically those with a high enough accuracy rating and can decide to pass, block, or delay suspicious user activity

Each rule evaluating the user request generates a rule result-
+1 = user request is from an attacker.
 0 = no decision.
-1 = user request is a human.

An aggregate score (a check result) for the user request is calculated as follows-

$$attacker\_rating = \frac{\sum^{All\ Rules} rule\_accuracy * rule\_result}{number\_rules\_applied} \quad (1)$$

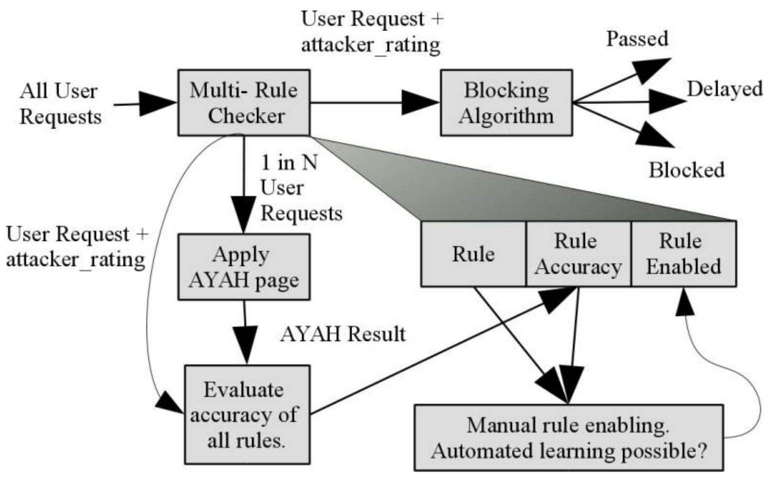


Figure 1 Real-time Scoring Process

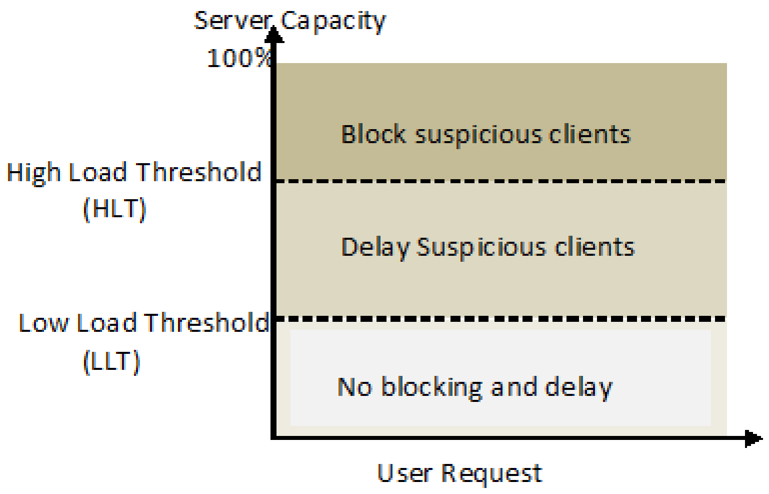


Figure 2 Server Load Limit

Given a user request gets a particular attacker_rating how does this translate into a decision on whether to block the user request or let it pass? From our work the ratings cluster around very high values or very low values so a threshold of 50 (half way between 0 and 100) works well. Future work will examine if it is possible to better set this value dynamically based on properties of the aggregate traffic. The decision to block, pass, or delay can be based on the attacker_rating and the total traffic being experience by the web server as shown in Figure 2. The delay method was not tested in our implementation and will be future work.

The algorithms developed satisfy the major objective of our work, in a dynamic environment not to punish (profitable) legitimate traffic while blocking the DDoS attacks. Without the real time calibration provided by the occasional AYAH page, this goal would not be achievable.

4 Experimental architecture

The real-time scoring algorithm has been implemented on an Apache web server where shared memory was used to allow very fast interaction between Apache and a Linklist Rule Daemon shown in Figure 3.

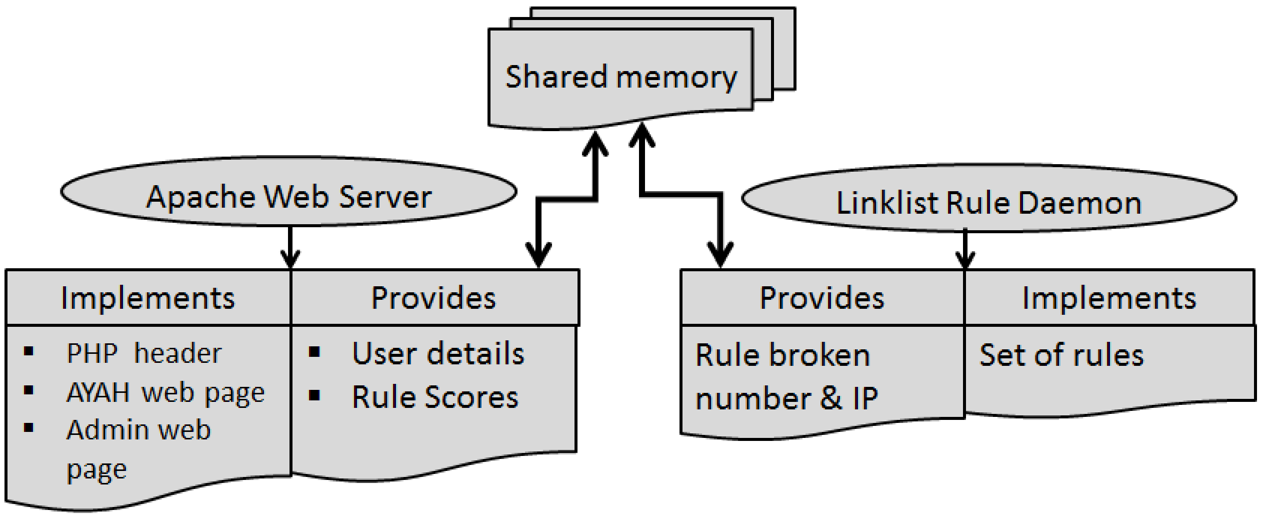


Figure 3 Experimental Architecture

4.1 Apache web server functions

The Apache web server plays a major role in discriminating the attack traffic from the legitimate users. Each web page requires a one line include statement which refers to PHP code that achieves the following-

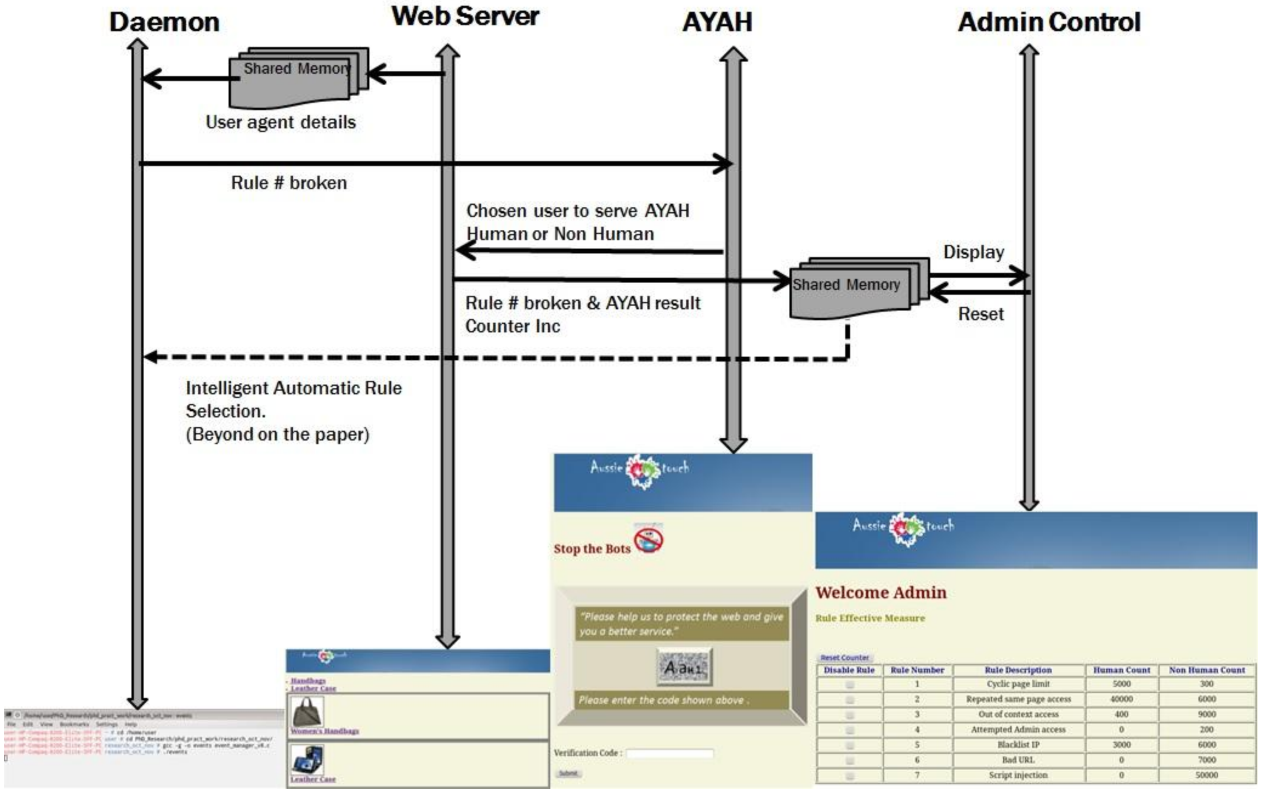
- Share client details.
- Share web page access details
- Serve AYAH web page if appropriate.
- Promoting or demoting a rule based on AYAH page result.

4.2 Linklist rule daemon functions

The Linklist Rule Daemon maintains a 2-dimensional link list based on IP address and time with time-based garbage collection. The daemon contains the rule engine which scans the 2D link list looking for rule violations.

Examples of rules include:

- Cyclic page limit: The client requests a number of similar page accesses per second. For example, an attacker may control the thousands of zombie systems programmed to make a continuous request for web pages http://page1 http://page2 http://page3. The daemon may set a limit for the pages per second allowed for any one client.
- Random limit: the user appears to be accessing pages at random.
- Repeated same page access: The client requests the same page again and again. The daemon may apply an initial burst limit and then a pages per second limit.
- Out of context access: The client accesses a resource such as video or audio outside a normal web page access.
- Attempted administration access: the server can report attempted accesses of the administration.
- Excessive access time: the user is accessing the website for much longer than normal users. This rule is not suitable for some websites such as online magazines and online news.
- Excessive repeated access: accessing the site for more than N sessions per period.



```
user-HP-Compaq-8200-Elite-SFF-PC ~ # cd /home/user
user-HP-Compaq-8200-Elite-SFF-PC user # cd PhD_Research/phd_pract_work/research_oct_nov/
user-HP-Compaq-8200-Elite-SFF-PC research_oct_nov # gcc -g -o events event_manager_v8.c
user-HP-Compaq-8200-Elite-SFF-PC research_oct_nov # ./events
```

Welcome Admin

Rule Effective Measure

Disable Rule	Rule Number	Rule Description	Human Count	Non Human Count
☐	1	Cyclic page limit	5000	300
☐	2	Repeated same page access	40000	6000
☐	3	Out of context access	400	9000
☐	4	Attempted Admin access	0	200
☐	5	Blacklist IP	3000	6000
☐	6	Bad URL	0	7000
☐	7	Script injection	0	50000

Figure 4 Experimental Results

4.3 AYAH operation

Figure 4 shows the key transactions between the system components and the web pages for the activity of evaluating rules (for the 1 in N pages). The user details of a request are sent from the Apache server to the daemon via shared memory and the daemon determines if any rules have been broken. The AYAH PHP code running on the web server then checks if the user is human and the result is passed into shared memory. An Admin Control web page can display the aggregated results of rule evaluation from the AYAH activity and under manual control mark a rule as active or inactive.

4.4 Experimental results

The implemented system was stressed with a variety of App-DDoS attack loads mixed with manual web accesses. It functioned perfectly allowing human requests to pass and blocking attack traffic that fell within the rule detection parameters.

To study the efficacy of the architecture with the shared memory and link list daemon we setup the IDS programs Snort and Bro to simply analyze HTTP traffic but not block it. Bro in particular is highly programmable and could be used to create some App-DDoS detection facility. Attack traffic of 7000 user requests per second was fired at these IDS systems, and the Apache –Link List implementation. On a 3 GHz PC Snort required 40% of a CPU core's time, Bro 96%, and the link list daemon only 3.3%. There is a direct relationship between CPU load and real power consumption [32] and so, while not the primary aim of this research, we can conclude that the daemon and shared memory architecture is not only effective but will also reduce real power consumption.

5 Future work

This paper has developed a novel method to block DDoS attacks on a web server and proved its viability in an experimental implementation. Much more research is yet to be done using this architecture. The approach needs to be used on a web site with significant attack traffic and real human flash traffic in order to test the implementation architecture and hone the basic rule set. In Figure 1 the rules that were active in blocking traffic were manually decided, based on the reported rule accuracy. This does not suit a 24/7 web farm where constant vigilance would be needed to inspect rule performance and move rules between active and inactive. An automated system with learning would be much better. There are interesting trade-offs between rule parameters, speed of traffic statistic changes, and accuracy with different types of traffic. Evaluating the efficacy of various learning methods would be very interesting research.

Another interesting feature is the threshold used for the attacker_rating which is used to declare a user request to be an attacker. This work used a value of 50 but this may not be an optimum value. Again a learning system may prove better and studying various learning methods with different traffic would be very interesting.

6 Conclusion

The work in this paper has described a novel architecture which can dynamically evaluate a set of rules for detecting and blocking Denial of Service attacks with minimal punishment of traffic originating from a human user. The key innovation is to use a very occasional CAPTCHAS (Are You a Human) page to test the effectiveness of a number of attacker detection rules. The successful rules are then applied to the entire traffic flow to block likely attackers. The basic architecture works well but for full production use learning algorithms need to be developed to dynamically enable and disable rules. This will be a very interesting area of future research.

References

[1]. X. Yi and Y. Shun-Zheng (2009), "Monitoring the Application-Layer DDoS Attacks for Popular Websites," *Networking, IEEE/ACM Transactions on*, vol. 17, pp. 15-25.

[2]. X. Yi and Y. Shun-Zheng (2009), "A Large-Scale Hidden Semi-Markov Model for Anomaly Detection on User Browsing Behaviors," *Networking, IEEE/ACM Transactions on*, vol. 17, pp. 54-65.

[3]. P. Park, H. Yi, S. Hong, and J. Ryu (2010), "An effective defense mechanism against DoS/DDoS attacks in flow-based routers," presented at the Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia, Paris, France.

[4]. X. Wang and M. K. Reiter (2004), "Mitigating bandwidth-exhaustion attacks using congestion puzzles," presented at the Proceedings of the 11th ACM conference on Computer and communications security, Washington DC, USA.

[5]. D. K. Y. Yau, J. C. S. Lui, F. Liang, and Y. Yam (2005), "Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles," *IEEE/ACM Trans. Netw.*, vol. 13, pp. 29-42.

[6]. R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker (2002), "Controlling high bandwidth aggregates in the network," *SIGCOMM Comput. Commun. Rev.*, vol. 32, pp. 62-73.

[7]. J. Haggerty, S. Qi, and M. Merabti (2005), "Early detection and prevention of denial-of-service attacks: a novel mechanism with propagated traced-back attack blocking," *Selected Areas in Communications, IEEE Journal on*, vol. 23, pp. 1994-2002.

[8]. R. Vaidyanathan, A. Ghosh, Y. Sawaya, and A. Kubota (2012), "On the use of Enhanced Bogon Lists (EBLs) to detect malicious traffic," in *Computing, Networking and Communications (ICNC), 2012 International Conference on*, pp. 1-6.

[9]. F. Soldo, K. Argyraki, and A. Markopoulou (2012), "Optimal Source-Based Filtering of Malicious Traffic," *Networking, IEEE/ACM Transactions on*, vol. 20, pp. 381-395.

[10]. J. B. D. Cabrera, L. Lewis, Q. Xinzhou, L. Wenke, R. K. Prasanth, B. Ravichandran, *et al.* (2001), "Proactive detection of distributed denial of service attacks using MIB traffic variables-a feasibility study," in *Integrated Network Management Proceedings, 2001 IEEE/IFIP International Symposium on*, pp. 609-622.

[11]. Y. Jian and K. Mills (2005), "Monitoring the macroscopic effect of DDoS flooding attacks," *Dependable and Secure Computing, IEEE Transactions on*, vol. 2, pp. 324-335.

[12]. J. Mirkovic, G. Prier, and P. Reiher (2002), "Attacking DDoS at the source," in *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*, pp. 312-321.

[13]. J. Mirkovic and P. Reiher, (2004) "A taxonomy of DDoS attack and DDoS defense mechanisms," *SIGCOMM Comput. Commun. Rev.*, vol. 34, pp. 39-53.

[14]. V. Durcekova, L. Schwartz, and N. Shahmehri (2012), "Sophisticated Denial of Service attacks aimed at application layer," in *ELEKTRO*, pp. 55-60.

[15]. S. Ranjan, R. Swaminathan, M. Uysal, A. Nucci, and E. Knightly (2009), "DDoS-Shield: DDoS-Resilient Scheduling to Counter Application Layer Attacks," *Networking, IEEE/ACM Transactions on*, vol. 17, pp. 26-39.

[16]. Y. Wei and L. Ming-Fang (2005), "Defending Application DDoS with Constraint Random Request Attacks," in *Communications, 2005 Asia-Pacific Conference on*, pp. 620-624.

[17]. J. Yu, C. Fang, L. Lu, and Z. Li (2010), "Mitigating application layer distributed denial of service attacks via effective trust management," *IET Communications*, vol. 4, pp. 1952-1962.

[18]. V. D. Katkar and D. S. Bhatia (2014), "Lightweight approach for detection of denial of service attacks using numeric to binary preprocessing," in *International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA)*, pp. 207-212.

[19]. V. D. Katkar and S. V. Kulkarni (2013), "Experiments on detection of Denial of Service attacks using Naive Bayesian classifier," in *International Conference on Green Computing, Communication and Conservation of Energy (ICGCE)*, pp. 725-730.

[20]. S. Zaman and F. Karray (2009), "Lightweight IDS Based on Features Selection and IDS Classification Scheme," *International Conference on Computational Science and Engineering*, pp. 365-370.

[21]. V. D. Katkar and D. S. Bhatia (2014), "Lightweight approach for detection of denial of service attacks using numeric to binary preprocessing," in *Circuits, Systems, Communication and Information Technology Applications (CSCITA), 2014 International Conference on*, pp. 207-212.

[22]. G. Wang, J. Hao, J. Mab, and L. Huang (2010), "A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering," *Expert Systems with Applications*, vol. 37, (September 2010).

[23]. C.-H. Tsang, S. Kwong, and H. Wang, "Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection," *Pattern Recognition*, vol. 40, pp. 2373–2391, (September 2007).

[24]. S. Kandula, D. Katabi, M. Jacob, and A. Berger (2005), "Botz-4-sale: surviving organized DDoS attacks that mimic flash crowds," presented at the Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2.

[25]. J. Rangasamy, D. Stebila, C. Boyd, J. Gonz, #225, and l. Nieto (2011), "An integrated approach to cryptographic mitigation of denial-of-service attacks," presented at the Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, Hong Kong, China.

[26]. D. Pogue. (2012), "Time to kill off Captchas," vol. ScientificAmerican.com, p.15,[Online] Available at: http://www.scientificamerican.com/article.cfm?id=time-to-kill-off-captchas (March 2012).

[27]. Cisco(2015), "Identifying and Mitigating the Distributed Denial of Service Attacks Targeting Financial Institutions", [Online] Available at: https://tools.cisco.com/security/center/viewAMBAlert.x?alertId=27115

[28]. Juniper(2012) "hit-rate-threshold", [Online] Available at: http://www.juniper.net/techpubs/en_US/junos12.1/topics/reference/configuration-statement/security-edit-hit-rate-threshold.html.(October 2012)

[29]. Juniper (2013), "IDP Application-Level DDoS Protection Overview." [Online] Available at: http://www.juniper.net/documentation/en_US/junos12.1x44/topics/concept/idp-application-level-ddos-protection-overview.html(January 2013)

[30]. Juniper(2014), "time-binding." [Online] Available at: http://www.juniper.net/documentation/en_US/junos12.1x47/topics/reference/configuration-statement/security-edit-time-binding.html (May 2014)

[31]. S. Sivabalan and P. J. Radcliffe (2013), "A novel framework to detect and block DDoS attack at the application layer," in *TENCON Spring Conference, 2013 IEEE*, 2013, pp. 578-582.

[32]. M. Weiser, B. Welch, A. Demers, and S. Shenker (1994), "Scheduling for reduced CPU energy," in *Mobile Computing*, ed: Springer, 1994, pp. 449-471.

[33].