IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 17, Issue 1, Ver. III (Jan – Feb. 2015), PP 54-61
www.iosrjournals.org

Preventing Web-Proxy Based DDoS using Request Sequence Frequency

Faizal N[1], Ramcy R [2], Anisha Thomas[3]
[1](Department of CSE, Younus College of Engineering, Kottarakara, Kerala, India. faizalnr@gmail.com)
[2](Department of CSE, Younus College of Engineering, Kottarakara, Kerala, India. vappichi14@gmail.com)
[3](Department of CSE, Younus College of Engineering, Kottarakara, Kerala, India. anithomas18@gmail.com)

Abstract: *In order to control the request flow in Computer Networks, a proxy server is used. Proxy Server is a server which acts as an intermediary server between server and clients. The more adaptable and converted attack is Web proxy-based HTTP attack in the existing DDoS attacks. In most of the large-scale official proxies which are usually configured to have high security sometimes cannot avoid being abused for the proxy base attacks. This type of attacks introduces new challenges to the existing security systems of the network. Web application is a set of technologies which serves for the web service. It is a good security practice to implement additional precautions to every mitigation level including the web application level for bringing the HTTP flood attack awareness for the web application. Here we introduce the detection and prevention of these flood attacks and DDoS attacks by using their frequency of request sequence arrival.*

Keywords: *Traffic analysis, Distributed Denial of Service attack, Distributed Security, Web Proxy.*

I. Introduction

A proxy server act as an intermediary server that may be a computer system or an application accepts requests from the clients. The proxy server accepts and evaluates the request from the client that connects to the server, in order to simplify and control its complexity. Client may request for the services like file, connection, web page, or other resource available from a server.

A web proxy may become an attacker by two steps:

- Attacker requests are collected by it and forced to forward the attack requests to the origin server;
- Disconnects the request flow between attacker and the proxy.

In step 1, in order to penetrate through the web proxies mainly two methods can be used: requesting dynamic documents or setting "Cache-Control: no-cache" in the headers of HTTP requests. A single host can simultaneously attack a lot of Web proxies to attack a Web server without the need of invading them by repeating these steps. The attraction of such an attack lies in three aspects:

- By connecting different Web proxies through HTTP protocols, the attacking host is enabled to break through the client-side restrictions.
- Due to lack of cooperation mechanisms between server and proxies, resisting such an attack by the mid Web proxies is not a practical approach, in particular those uncontrollable private proxies;
- Such an attack may confuse most of the existing detection systems designed for the traditional DDoS attacks due to two reasons: first, the terminal hosts shielded by the hierarchical proxy system are not directly observed and diagnosed by the origin server; second, the attack traffic is mixed with the regular client-to-proxy traffic by each proxy that forwards the traffic.

In the final aggregated proxy-to-server traffic, except their underlying purposes, there is no obvious difference between the normal traffic and the attack traffic. So it is hard to identify and filter the attack requests accurately by the victim server.

The Web proxy-based HTTP attack is more flexible and private than most of existing DDoS attacks. Mainly difficulty of detection lies in three aspects:

- Real attacking hosts are protected using the hierarchical Web proxies, and so they are unobservable to the origin serve;
- A Web proxy may be inactively involved in an attack event and may unconsciously act as an attacker;
- From the victim server, both legal and illegal traffic are observed as they come from the same sources (i.e., Web proxies).

Even though most of the large-scale official proxies are usually configured for acquiring high security, the abusing of the proxy base attacks cannot be avoided. This brings new challenges to the proper functioning of the existing network security systems. A novel resisting scheme is proposed by the motivation of these issues

to protect the origin server from Web proxy-based HTTP attacks. The proposed scheme is based on network behavior analysis. It maps a Web proxy's access behavior to a hidden semi-Markov model (HsMM) which is a typical double stochastic processes model. The observable varying process of a proxy-to-server traffic is the output process of an HsMM profiles. The hidden semi-Markov chain of an HsMM transforms internal behavior states of a proxy and this can be considered as the primitive driving mechanism of a proxy to server traffic. By measuring the model deviation between an observed behavior and the Web proxy's historical behavior profile, the abnormality of a web proxy based on the above behavior can be detected by. Short-term and Long-term behavior assessment methods are proposed. The abnormal request sequences embedded in the proxy-to-server traffic is the short-term behavior and the issues warnings on a large scale is the long-term behavior. A new "soft-control" scheme, which reshapes the suspicious sequences from a source according to the proxy's historical behavioral profile, is developed for getting attack response. The system doesn't deny the entire sequence but partly discards the most likely malicious requests from a suspicious sequence considering the sequence as normal one. It helps the HTTP requests of authorized users to the greatest extent which avoids the possibility of request being discarded.

II. About the attack

In order to accomplish a denial of service state on systems, flood attacks aim to push boundary of system usage to the out of limit is determined by the normal traffic usage scenarios. There may have flood attack between the considered normal traffic and the considered abnormal traffic. The flood attack name can be determined by the specific protocol that attack is made on. For example, DNS Flood attack is the flood attack on the DNS protocol and e a flood attack is the HTTP protocol is known as HTTP Flood Attack. As every protocol has its own technical architecture and vulnerabilities, these flood attacks can differ on the attacking techniques from protocol to protocol, a type of flood attack.

2.1 Application DDoS Attacks

Application DDoS attacks are DDoS attacks targeted at powerful areas such as Web server, database resources, application server etc. The application layer based flood attacks still only account for ¼th of all DDoS attacks and they are more complicated and much more challenging to stop. Because the attack traffic often act as regular traffic and cannot be identified by network layer anomalies and so the application DDoS attacks usually bypass most traditional network security devices. Some application DDoS attacks simply flood a Web application with permissible requests in an attempt to immense processing power. Other attacks may exploit the business logic flaws. For example, search mechanism in a Website may require excessive processing by a back end database server and it becomes the main target for the attackers. By performing thousands of search requests using wildcard search terms to overwhelm the back end application database an application DDoS attack could exploit this weakness."Slowloris" emerged as a more dangerous application based DDoS attack reported in 2009. The "Slowloris" disrupts application service by exhausting web server connection pools. In this attack, the incomplete HTTP request send by an attacker along with periodic header lines to keep the connection alive, but never sends the full request. An attacker can open numerous connections and overwhelm the targeted Web server without requiring much bandwidth. While multiple patches have been created for Apache and other web servers to mitigate this vulnerability, apart from that it demonstrates the power of more complicated DDoS attacks. Denial-of-Service (DoS) attacks continued as the key threat in internet based applications. In the DDoS attacks, a team of attackers generates a large amount of traffic, soaked with the victims network, and causes remarkable damage. The overlay networks, known as proxy networks must be proposed to protect applications against such DoS attacks. Using the proxy network to mediate all communication between users and the application, thereby it prevents the direct attacks on the application layer. This is the key idea is to hide the application behind a proxy network. Realistic study of these approaches should involve large networks, real time applications, and real time attacks. However the study of these perspective have been limited to theoretical analysis and small-scale experiments, which cannot capture the complex system dynamics, including dropping packets, router queuing, temporal and feedback based behavior of network and application based protocols during DoS attacks. To the application and proxy network performance in the face of DoS attacks these factors are critical. Thus, we still do not have answers to many key questions about the viability and properties of these proxy approaches. Can proxy networks tolerate DoS attacks, particularly, with real and complex network structures and protocol behavior? If so, what are the key parameters to achieve effective and efficient resilience? What are the performance implications for applications of using proxy networks? Our main contributions are the following.

- we provide the first large-scale empirical study on the DoS resilience capability of proxy networks using real applications and real attacks; this is a qualitative advance over previous studies based on theoretical models and small scale experiments.
- We provide the first set of empirical evidence on large-scale network environment to prove that proxy networks have effective and scalable resilience against DoS attacks.

- Detailed performance analyses of proxy networks in large-scale network environment is provided, and show that, in contrast to intuition, proxy networks can improve user-experienced performance.

III. Problem Statement

A typical Internet application deployment, such as an e-Commerce application can be emphasized in Figure 3.1. On a cluster of servers, usually the application services are running. Users are distributed across the Internet, through which it access the application service. As shown in Figure 3.2, in this application model, the Internet plays a communication layer, which conveys the well-defined application-level protocol between the applications and users. Examples of such applications include search-engines, online banking, e-commerce and e-trading applications. The applications depend, making the applications unavailable to their users.

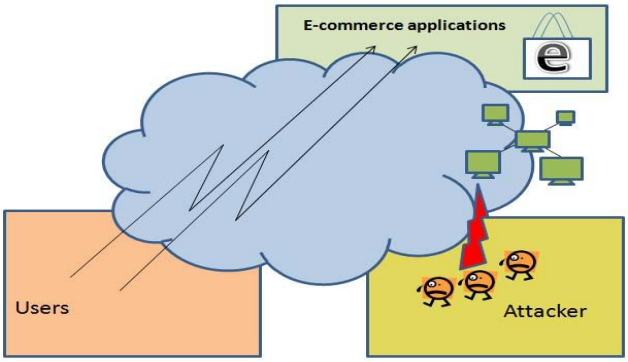

Figure 3.1 Internet Applications & DoS Attacks

There are two classes of DoS attacks: infrastructure level and application-level attacks. Infrastructure-level attacks are attacks that directly attack the resources of the service infrastructure. Infrastructure includes the networks and hosts of the application services. For example, to saturate the target network, attackers send floods of network traffic. Attacks through the application interface are another one is called application-level attacks. For example, for sending offensive workload, attackers may overload an application, or malicious requests to the application which crash the application, attackers can overload an application.

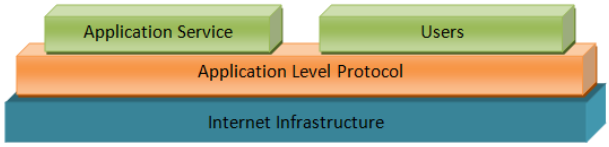

Fig 3.2 Application model

Infrastructure-level DoS attacks only require the knowledge of applications network address. Application-level DoS attacks are application-specific so they do not require the target application's IP address. Distributed Denial-of-Service (DDoS) attacks are large-scale DoS attacks which employ a large number of attackers distributed across the network. Mainly there are two stages in attacks.

- A large zombie networks are being built by the attackers compromising several Internet hosts by installing a zombie program on each network.
- Attackers activate this large zombie network, directing them to a 'DoS' target.

In stage two, both infrastructure and application-level DoS attacks can be used. Automated DDoS toolkits, such as Trinoo, TFN2k and mstream, and worms, such as CodeRed, provide automation, enabling large scale attacks to be easily constructed.

1.1. Proxy Network Approach

To protect applications from DoS attacks proxy networks have been proposed. A generic proxy network encompassing most of the proposed approaches is illustrated in Figure 3.2. As shown in Figure 3.3, an overlay network, or a proxy network, is used to mediate all communication between users and the application. The proxy network is the only public interface for the application on which the mediation can be enforced and the application cannot be directly attacked. Edge proxies, a large set of proxies publish their IP addresses, providing application access. We can flexibly increase the number of edge proxies. Scalable resilience against DoS attacks on edge proxies allowed by this, and thereby allows a proxy network to shield the application from

DoS attacks. We study the fundamental capabilities and limitations of a wide range of proxy networks using this generic proxy network model.

As discussed above, to successfully protect applications from DoS attacks a proxy network must have the following two key capabilities.

- A proxy network must enforce mediation so that the application can only be reached via the proxy network, thereby preventing direct DoS attacks on the application.

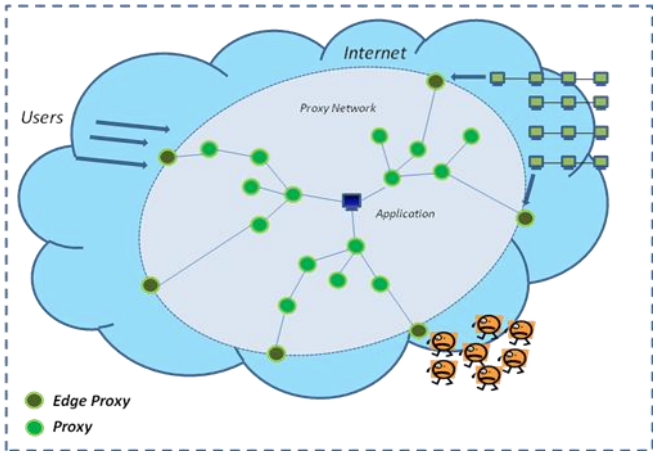

Figure 3.3 Proxy server approach

- Second, a proxy network must provide DoS-resilience mediation so that it can support continued user access to the application under DoS attacks.

Using a proxy network, most of the Proxy server doesn't detect or resist these DOS attack and they also hide an application IP address, thereby enforcing proxy network mediation. Thus the web server cannot group request from each application. So most of the HTTP resisting algorithms are grouped requests on proxy server ID based. Our base paper uses this approach.

To resist the Web proxy-based distributed denial of service attack a novel server-side defense scheme is proposed. To extract the behavior features of the proxy-to-server traffic, this approach utilizes the temporal and spatial locality, which makes the scheme independent of the traffic intensity and frequently varying Web contents. To protect weak signals from the interference of infrequent large values a nonlinear mapping function is introduced. Then, a new hidden semi- To describe the time-varying traffic behavior of Web proxies, Markov model parameterized by Gaussian-mixture and Gamma distributions is proposed. The number of parameters to be estimated is reduced by the new method, and can characterize the dynamic evolution of the proxy-to-server traffic rather than the static statistics. To meet the requirement of both fine-grained and coarse-grained detection, two diagnosis approaches at different scales are introduced. Soft control is a novel attack response method proposed in this work. Suspicious traffic is converted into a relatively normal one by behavior reshaping rather than rudely discarding.

1.2. Existing system

To extract the proxy to server behavior, temporal and spatial locality analysis used. Temporal locality of reference has been widely applied in many fields. For example, program behavior reference pattern of Web access and Web proxy cache replacement approach. Temporal locality refers to the referencing behavior of a client in the recent past can be used to predict the referencing behavior of the same in the near future. The frequency of the requests without indicating the correlation between a reference to a document and the time since it was last accessed can only be represented by the resource popularity metric. Here, the concept of stack distance is resorted. The files are assumed to be placed on a stack such that, whenever a file f is requested, it is either reposition to the top of the stack, or simply pushed to the stack if the file is not found in the stack. The distance of f from the top in the former case is the stack distance or undefined in the latter case.

Objects neighboring an object frequently accessed in the past are likely to be accessed in the future; this property is referred to as spatial locality. For example, when a home page is requested, all its embedded objects are likely to be accessed at the same time. Thus the spatial locality indicates correlation among a cluster of HTTP requests, capturing spatial locality can help to the access the behavior of Web proxies. Different from, here a new method is used to quantize the spatial locality.

The system uses hidden semi-Markov model, model without state information. That is requests are grouped based on proxy server ID. This model has no idea about original request source. Because proxy server

hide this information from web server due to the HTTP protocol limitation. We can detect both temporal and spatial behavior in this system by using this model. Instead of blocking proxy server requests from attackers and non attackers through proxy server are reached at web server side in a mixed manner, this model reshape incoming request to remove attacking request. To split this requests, the Web server have no facility. From this request, web server detects spatial and temporal behavior using this Markove model. But, when number of non attacking client increases, this will increase false positive and false negative ratio.

IV. Our Proposed Solution

The vulnerability in the protocol is the main reason of flood attacks. For example, a UDP Flood or SYN Flood attack uses the nature of protocol's design to saturate the network traffic. In a SYN Flood attack, attacker uses the TCP 3 way handshake's first initiation step to spoof IP addresses and to drain server side system/network resources. Attacker uses the stateless design of the UDP protocol to spoof IP addresses and to drain server side system and network resources, as it comes to the UDP Flood attack. For this reason, every mitigation method for the flood attacks must be implemented in a consideration and perspective of system/protocol design to accomplish an effective security solution. It is possible to detect and analyze packet payloads only by application layer security devices like IPS or WAF (Web Application Firewall), since HTTP protocol serves at the application (7^{th}) layer of the OSI Model. There are no inspection and analyzing chance on the HTTP flood attacks for other security devices which do not serve at the application (7^{th}) layer. The only detection way for these devices is TCP connection counts made for the HTTP responses. As a result of detection, HTTP Flood attack attempts can be prevented by and blocked on different layers of OSI model other than application (7^{th}) layer.

In the real world scenarios, there are many situations that the HTTP flood attacks are not mitigated properly. Some of them might be related with the security configuration weakness or the absence of a security device. Situations like these might be handled with the other security enhancements at the different level of the information technology architecture. This is where the web application level comes in. An application-layer solution works within the application that it is protecting, unlike the network-layer protection products. Web application is a bunch of technologies which serves for the web service. It is important to clarify whether the attack is a HTTP flood attack or not, before starting a discussion on the web application level approach to the HTTP flood attacks. If a TCP packet carrying an HTTP request payload should be interpreted by the web service, then attack attempt is considered as an HTTP flood attack . In HTTP flood attack, the attack surface always begins with the web service and its backend infrastructure. A HTTP flood attack attempt is just a TCP DoS attack that saturates the network traffic and so which cannot make it to the web service.

To create a resistance at the web application level against the HTTP flood attacks, the basic idea might be summarized into 3 steps:

a. Detect IP addresses of the abnormally excessive requests based on the normalized traffic.
b. To reduce attack surface, reply to these requests with a low resources like blank page.
c. Block detected IP clients by using other components at the other levels of mitigation (WAF, web server/service, etc.).

This implementation will also save resources of the backend infrastructures like distributed services, SQL Servers or e-mail/application/media servers, etc. while reducing attack surface by sending low resource used responses. For the network architectures which share the backend infrastructure members with other infrastructures like intranet or distributed web application servers, this is extremely important and critical. We can reduce the amplitude of the HTTP flood attacks by saving the resources for the backend infrastructure.

It is a good security practice to bring the HTTP flood attack awareness for the web application and implement additional precautions to every mitigation level including the web application level. It can easy to detect this attack following rule.

1.3. The rule creation

For the web application level HTTP flood attack mitigation, the false positives are considered as the critical point. In order to avoid false positives in the detection, the rules must be well defined and be tested with the real world traffic usage scenarios. Also a good understanding for the rule creation concept is highly suggested. In the proxy server an enhanced system of HTTP protocol is implemented here. So proxy server doesn't hide application ID from web server. Thus web server got client identity of each request, and also the client can group requests based on this application ID.

Our modified algorithm

- Include a "DBMS:: Flat File" or defined RAM area save all action and reaction logs to it.
- Record request arrival time in microseconds and its count alongside with the IP addresses.
- Compare the counts and times of each and every requests from the same IP address.

- Record the IP addresses and time values upon a rule breach.
- For white listed IP addresses some exception rules are defined.
- Record every IP that breached the rule.
- A pseudo code "EXIT" is used to stop web application execution to reduce attack surface upon a breach.
- Define a time limit for the suspension of the web application execution,
- Send detected IP addresses to the other security components (e.g. stateless firewall)

The first step should be the defining of the normal traffic, in order to accomplish a standardized rule base against HTTP flood attacks.
The normal values for the network traffic on web application as follows:

- Maximum requests from a single IP address per unit time will be 5,
- The time between the closest two requests from a single IP address - 0.2 seconds.

These are the baseline standardized traffic values for creating a rule against the HTTP flood attacks. To create a healthy rule, this is the minimal information to create.

A basic abnormal traffic rule set on the baseline sample value "number of request in 0.1 seconds is 10". According to the normal traffic baseline rule set, 1 request in 40.000 microseconds from an IP address will not be considered as a HTTP flood attack. From a single IP address, the abnormal traffic rule above allows 1 request in 10.000 microseconds at 10 times. According to the rule creation concept, this rule also has a tolerance factor pointed out by 10 times the description. To mitigate false positives, the tolerance factor (the request count) can give an opportunity. This web application level implementation can provide an opportunity to slow down the Brute Force Attacks and Web Vulnerability Scanners besides HTTP flood attacks. When the detected IP address shared with other security components, to provide an opportunity to block detected IP, the attacker. We also implement different mechanism for detecting following attacks, in addition to this traffic rule

1. Unsupported HTTP method
2. Oversized Header and Body data size
3. Large or small time out interval
4. Minimum incoming data
5. SQL Injection
6. Command Execution (through CGI scripts).

By packet analysis technique, these attacks can be detected. We implement a packet capturing and analysis tool for this.

1.4. Modules

1.4.1. Proxy Server: - Develop a proxy server which acts as proxy for client. This server interconnect client and web server. The clients seeking resources from other services send requests to an intermediary server that may be a computer system or an application, such a server is called a proxy server. A client connects to the proxy server requesting some service, such as a file, connection, web page, or other resource available from different servers and the proxy server evaluates the request as a way to simplify and control its complexity. Proxies are invented to add structure and encapsulation to distributed systems.

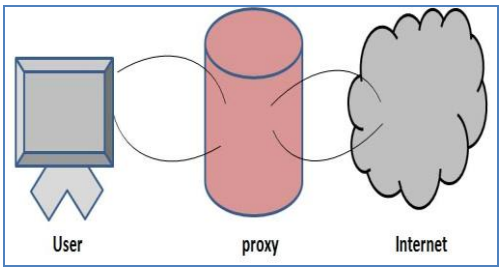

Figure.4.2 Reading web request

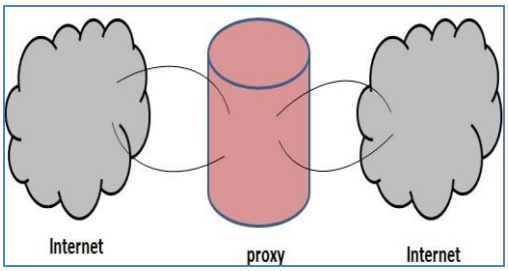

Figure. 4.1 Sending request to server

Different Steps

i. Create Server Socket for accepting connection from browser.
ii. Accept connection.
iii. Read web request
iv. Establishing connection to web server.
v. Send Request to Web Server.
vi. Read response from server.

vii. Send Response to client.

1.4.2. Web server: - Develop an HTTP Server. This contain following modules

1.4.2.1. **HTTP Handler :-** Develop a client server program for processing HTTP request

a. Create a server socket listen to port 80
b. Read Request.
c. Separate url and cgi parameters.
d. Create response.
e. Combine response and header.
f. Send to client.

4.2.2.2. **Training :-** find different request pattern (spatial and temporal) , develop Training Model

4.2.2.3. **Detecting: -** develop a system for detecting abnormal request pattern.

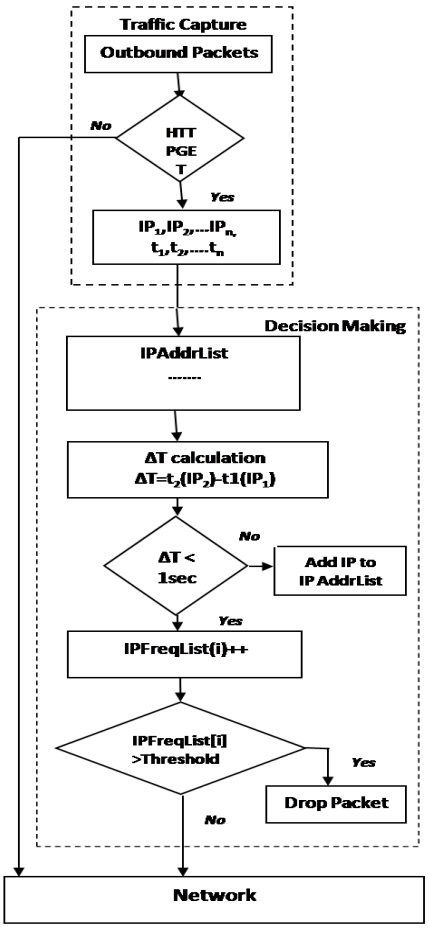

Figure.4.3 RSF Based detection flowchart

Here we implement a new algorithm (Request Sequence Frequency Based Attack Detection – RSFBAD) for detecting attacks modules of the proposed RSFBAD, i.e., traffic capture, parameter extractor, and the analyzer module. First the packets are captured at kernel level by the traffic capture module. The output of the traffic capture module, outbound TCP packets alone, are filtered and sent to the parameter extractor module which extracts the features such as remote IP address and arrival time of packets. Then these packets are subjected to the analyzer. Based on the frequency of request sequence set, the analyzer decides whether to drop or allow the packets into the network.

The analyzer module contains an IP frequency list used to store the number of request occurrences from individual IP address over a period of time. It checks the frequency of each IP address in IP frequency list and decides whether to allow or block the packets received. The flow chart for the basic functioning of RSFBAD is given in Figure.4.3. The number of HTTP requests to a particular IP address is restricted by setting normalized frequency values for a given period of time which is predefined. Based on the parameters extracted

from the packets ΔT values are calculated, which gives the time interval between current and previous instance of a packet for a particular IP address. If the value of the IP frequency list exceeds the normalized frequency value, the packets are dropped. This time interval calculation is done based on the threshold set.

The IP frequency list is updated based on the presence of the extracted IP address and checks whether it is in the IP address list. If the searched IP address is not found in the IP address list, then the new IP address and its corresponding arrival time are stored in the IP address list. Then in Step 3, the difference in time (interval) between two packets of same IP address (ΔT) is calculated. If the time interval is greater than or equal to 1 second, then the IP address is added to the IP frequency list. Otherwise the IP frequency list value for the corresponding IP address is incremented by 1.

The refresh time interval to the list is set to 1 second and also the IP frequency list values are reset to null after the elapse of every second. As per the experiments conducted threshold value (N), that is the maximum frequency count is set to 20, that means., only 20 HTTP GET requests are allowed to a particular IP in one second. Until the corresponding IP frequency list value reaches N, the HTTP GET packets are allowed to an IP address., Then the packets are dropped, if the IP frequency list value for an IP address exceeds N.

V. Conclusion And Future Work

The application layer attack has become a major threat to the internet in nowadays. The focus of this paper is to come out with an effective solution for the detection and prevention of clients from inadvertently taking part in such attacks. Accordingly, to prevent the DDOS attack, Request Sequence Frequency based attack detection Based HTTP Filter (RSFBHF) was proposed and implemented. It is easy to identify the request from the attacker and so deny the request within the proxy server without interrupting the server system by analyzing the temporal and spacial locality behavior of any requested client. We tried to detect the attack sequence using the spacial and temporal locality behavior and the request sequence frequency. Threshold based frequency detection system may include some false negative and false positive counts along with the result. In order to improve the functionality, the algorithm can be modified based on the training, normal request rates, frequency of sequence etc. Next level enhancement is possible to improve based on the following improvements to the system developed.

- Show a CAPTCHA based entry to user who is accidently blacklisted and want to avoid waiting for a suspension time whenever an attack found.
- Refinement of algorithm for reducing the percentage of false positive and false negative error.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of this paper. They also thank Computer Science department of Younus College of Engineering, Kottarakara, Kerala, for assisting them to collect the real experiment data reported here. They are also thankful to the support given by the management and staffs of Younus College of Engineering. They specially thank Mr. Monilal S, Lecturer, department of CSE, Govt. Polytechnic College, Kayamkulam, for his valuable support in technical development for this research done by Mr. Faizal N.

References

[1]. Yi Xie, S. Tang, , Y. Xiang, and J. Hu, "Resisting Web Proxy-Based HTTP attacksby Temporal and Spatial Locality Behavior" IEEE Transactions On Parallel And Distributed Systems, Vol. 24, No. 7, July 2013

[2]. SMs P.Rajani Reddy, Mr R.Siva and Ms C.Malathi "Techniques to Differentiate DDOS Attacks from Flash Crowd" International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 6, June 2013.

[3]. LI Jin-ling, WANG Bin-qiang, "Detecting App-DDoS Attacks Based on Marking Access and d-SVDD" in 2nd International Symposium on Computer, Communication, Control and Automation (ISCCCA-13) ,2013.

[4]. Mohamed Ibrahim AK and Lijo George, Kritika Govind and S. Selvakumar, "Threshold Based Kernel Level HTTP Filter (TBHF) for DDoS Mitigation".: I.J. Computer Network and Information Security, 2012, 12, 31-39 .

[5]. Mohammed Alenezi, Martin J Reed, "Methodologies for detecting DoS/DDoS attacks against network servers".: The Seventh International Conference on Systems and Networks Communications, ICSNC Semi-Markov models 2012.

[6]. Davide Ariu, Roberto Tronci, Giorgio Giacinto , "HMMPayl: An intrusion detection system based onden Markov Models" Elsevier December 2010 and computers & security magazine volume 30 (2011).

[7]. Igino Corona,Giorgio Giacinto, "Detection of Server-side Web Attacks" JMLR: Workshop and Conference Proceedings 11 (2010) 160{166.

[8]. Shun-Zheng Yu, "Hidden Semi-Markov models". Elsevier, 14 April 2009.

[9]. Peng, T., Leckie, C., and Ramamohanarao, "Survey of network-based defense mechanisms countering the DoS and DDoS problems." , ACM Computer. Survey. 39, 1, Article3, April 2007.

[10]. Frank Kargl, Joern Maier, Michael Weber, "Protecting Web Servers from Distributed Denial of Service Attacks" May 1-5, 2001, Hong Kong. ACM 1-58113-348-0/01/0005.

[11]. R Sekhar, M Bendre, D Dhurjati "A Fast Automation Based Method For Detecting Anomalous Program Behaviour," 0-7695-1046-9© IEEE 2001.

[12]. Tao Peng, Christopher Leckie and Kotagiri Ramamohanarao "Survey of Network based defense mechanisms countering the DoS and DDoS problems" ,0-7695-1046-9© IEEE 2001.