Volume : 3 | Issue : 5 | May 2014 • ISSN No 2277 - 8179

Research Paper

A Novel Approach For Resisting Web Proxy Attacks By Using TSL Behavior

Engineering

KEYWORDS : HSMM, HTTP attack, Soft control, Web proxy

Leo Elsa Koshy	PG Scholar, Dept of CSE, Vedavyasa Institute of Technology, University of Calicut, Kerala, India
Libi Babu	PG Scholar, Dept of CSE, TKM Institute Of Technology, CUSAT, Kerala, India
Suvya.P	Assistant Professor, Dept of CSE, Vedavyasa Institute of Technology, University of Calicut, Kerala, India

ABSTRACT

Web proxy based HTTP attack has been a serious threat to server security due to the covert nature. Although most of the large scale official proxies are usually configured to have high security, they cannot avoid being abused for the proxy based attacks. Such type of attack is not easy to be discovered by most existing defense systems since the real attacking hosts are shielded by the hierarchical Web proxies. In this paper a novel server-side defense scheme is proposed to resist such covert indirect attacks. This approach extracts the temporal and spatial locality behavior of the proxy to server traffic independent of traffic intensity and frequently varying Web content. Then, Web proxy's access behavior can be directly mapped into a HSMM parameterized by Gaussian mixture and Gamma distributions. A novel attack response method named soft control is proposed. This scheme reshapes the suspicious sequence into a relatively normal one by partly discarding its most likely malicious requests instead of denying the entire sequence. So it can protect the quality of service of users. Finally, experiments are conducted to validate the model.

INTRODUCTION

This paper proposes a novel approach for resisting web proxy attack. Web proxy attacks are a serious critical threat to the Internet application. It may cause Denial of Service in Internet application. So this paper mainly aims to avoid the Denial of Service of different Internet applications provided by the network. Only few studies have paid attention to this issue, currently. So most of the attackers concentrates on this type of attack.

A proxy server may be a system or computer application. It acts as an intermediary between a client and server. Proxy server evaluates the request as a way to simplify and control its complexity. Web proxies are kind of proxy that facilitating access to content on World Wide Web. Web proxy receives requests from different clients connecting to the web proxy and forward that requests in a combined manner to the web server. Web-server receive the requests and provide responses. The web proxy returns back their responses to each client.

Here, the client which acts as attacker sends a flood of requests at a time. And after sending the requests the attacker disconnects the connection between the attacker and web proxy. Due to the presence of these excess requests, increasing the load on web server and happen denial of service. So the non-attacking clients do not get responses correctly. Denial of Service problem is nothing but denying the services of clients. The Fig. 1 shows the connection provided by the proxy server between a client and server.

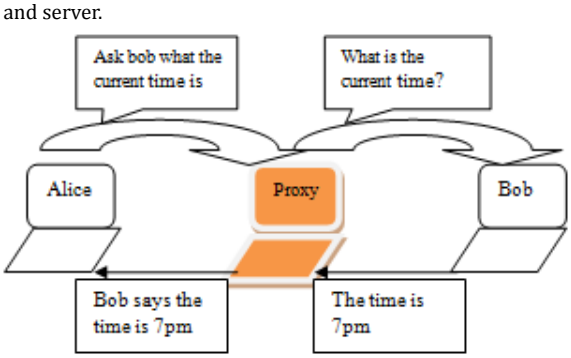


Fig. 1 Proxy Server

A proxy network must have two key capabilities to successfully protect applications from DoS attacks. First, 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. Second, a proxy network must provide DoS-resilience mediation so that it can support continued user access to the application under DoS attacks. But most of the Proxy server doesn't detect or resist these DoS attack and they also hide an application's IP address using a proxy network, thereby enforcing proxy network mediation. So webserver 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.

The web proxy attacks are difficult to find. Because the web proxies forward the requests to web server after hiding the details of client. So the web server can't find which client sends the flood requests and prevent the attacker. The web server can only take actions to the innocent web proxy. If it denying the services of that particular web proxy that will also affects the non-attackers.

Based on the network behavior analysis here proposes a novel scheme for resisting the web proxy based attacks. It extracts the network behavior in the form of temporal and spatial locality feature and maps it into HsMM. Based on this this behavior model the abnormality of further requests can be find out. Calculating a checksum value and so can detect the clients which sending each requests. And implement a new web proxy that does not hide the details of clients to the web server. That means an enhanced version of http protocol that provide the IP address of client is produced by the web proxy.

PROPOSED SYSTEM

The system tries to find out the attack and prevent the webserver from being the attack. It uses the Temporal and Spatial Locality (TSL) model of the request pattern. It provides a GGHsMM by using this locality model. It trains the web server system by extracting the TSL model of different request pattern and so the behavior and structure of these request patterns. During this training the system provides some threshold values for the request sequence. Then further request patterns are evaluated by using these threshold values and find out the attacking and non-attacking requests. This system improved by providing novel approach for identifying the sender of each request. Two approaches are used for this: checksum calculation and implementing an enhanced http protocol. A soft control scheme is established for attack responses. This soft-control scheme prevents the responses of attacking requests but provides response for non-attacking requests.

Temporal Locality

Temporal locality[1] was a property of a request in the request

pattern. This property deals with the referencing behavior of the requests. It means that a request recently accessed in the recent past may be accessed in the near future. That is, it deals with the requests which are repeated almost the same time.

For extracting the temporal locality feature this paper implements a stack distance model. In it the requests are enclosed in a stack. In the evaluation phase the requests in the stack are evaluated and if the distance between the same requests in the stack are lower than the threshold value and that particular request repeated over a certain value then that request was considered as an attacker.

Spatial Locality
Spatial locality[2] was a property of a sequence of requests in the request pattern. This property deals with a request sequence that appears repeatedly in the request pattern. It prevents the probability of the attack that attacker providing the attack by sending a cluster of requests repeatedly.

Here use joint entropy concept for extracting the spatial locality behavior. In this method, keep a copy of each request and when the next request came make a copy of the combination of current and previous request only if the that request can be appear after the previous request in the network application model. Then repeating this step based on the arrival of each request in the request pattern. And these copies are considered as request sequences. When a request sequence appears more than the threshold value, that sequence is considered as an attack.

GGHsMM
The GGHsMM means that Hidden semi-Markov Model[3] with Gaussian and Gamma distribution. Here this model is used for representing the extracted temporal and spatial locality behavior. The main advantage for using the GGHsMM is the reduction of number of parameters and so it will reduce the computational complexity.

Checksum Calculation
With every http request[4] some details of the resource should passed. By using these details, calculate a checksum value for each request. The checksum value will be different for each request. Implement a rule concept for limiting the requests from attackers.

Web Proxy Enhancement
The web proxy improved by implementing an enhanced http protocol. The enhanced http protocol reveals the ip address of client to web server. So the webserver can detect the client identity of attacking and non-attacking requests.

III. SYSTEM IMPLEMENATATION

Web application is a bunch of technologies which serves for the web service. Before starting a discussion on the web application level approach to the HTTP flood attacks, it is important to clarify whether the attack is a HTTP flood attack or not. To consider an attack attempt as a HTTP flood attack, a TCP packet which carries a HTTP request payload should be interpreted by the web service. Attack surface for HTTP flood attacks always begins with the web service and its backend infrastructure. A HTTP flood attack attempt, which cannot make it to the web service, is just a TCP DoS attack that saturates the network traffic.

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

- detect ip addresses of the abnormally excessive requests according to a previously defined rule,
- to reduce attack surface, return these requests with a low resource used response (like a blank page or else),
- block detected IP addresses

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:

3.1 Rule Creation Concept

The critical point for the web application level HTTP flood attack mitigation is the false positives. In order to avoid false positives, the detection rules must be well defined and be tested with the real world traffic usage scenarios. Here implement an enhanced HTTP protocol in this proxy server. So proxy server doesn't hide application id from webserver. So web server got client identity of each request. So client can group requests based on this application ID.

Modified Algorithm:

- save all logs to "DBMS:: Flat File" or RAM,
- record the every request time microseconds and counts alongside with the IP addresses,
- compare the counts and times of every requests made by the same IP address,
- record the IP addresses and time values upon a rule breach,
- create exceptions for white listed IP addresses,
- record every IP addresses that breached the rule,
- stop web application execution to reduce attack surface upon a breach with the pseudo code "EXIT",
- define a time limit for the suspension of the web application execution,
- send detected IP addresses to the other security components (eg. stateless firewall)

In order to accomplish a healthy rule base against HTTP flood attacks, the first step should be the defining the normal traffic.

The normal network traffic values on the web application are given below:

- maximum request count from a single IP address: 5 requests/second,
- time between the closest two requests from a single IP address: 0.2 seconds.

These are the baseline traffic values for creating a rule against the HTTP flood attacks. This is the minimal information to create a healthy rule. A basic abnormal traffic rule based on these baseline values could be sampled as "10 requests in 0.1 seconds". According to the normal traffic baseline values, 1 request in 40.000 microseconds from a single IP address will not be considered as a HTTP flood attack. The abnormal traffic rule above allows 1 request in 10.000 microseconds at 10 times from a single IP address. According to the rule creation concept, this rule also has a tolerance factor pointed out by "10 times" description. The tolerance factor (the request count) can give an opportunity to mitigate false positives.

In addition to the enhanced http protocol we also implement an another technique for identifying the sender of each request. Calculate a checksum value, ΔT by using the details provided by the http requests. All http requests should provide some informations like browser, operating system, etc. The MD5 algorithm is used for calculating the checksum.

3.2 Threshold Based Attack Detection

Here implement a new algorithm (Threshold based attack detection – TBAD) for detecting attacks modules of the proposed TBAD, viz., 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, the outbound TCP packets alone, are filtered and sent to the parameter extractor module which extracts the features such as remote IP address and the arrival time of packets. Then the packets are subjected to the Analyzer which decides whether to drop or allow the packets into the network based on the threshold set.

The analyzer module contains an IP frequency list to store the number of occurrences of individual IP address over a period of time. It checks the frequency of each IP address in IP frequency list and further decides whether to block or allow the pack-

ets. The flow chart for the basic functioning of TBHF is given in Fig. 2. Threshold value is set to restrict the number of HTTP requests to a particular IP address for a given period of time. 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. Based on the threshold set if the value of the IP frequency list exceeds the threshold value, then the packets are dropped else they are allowed into the network.

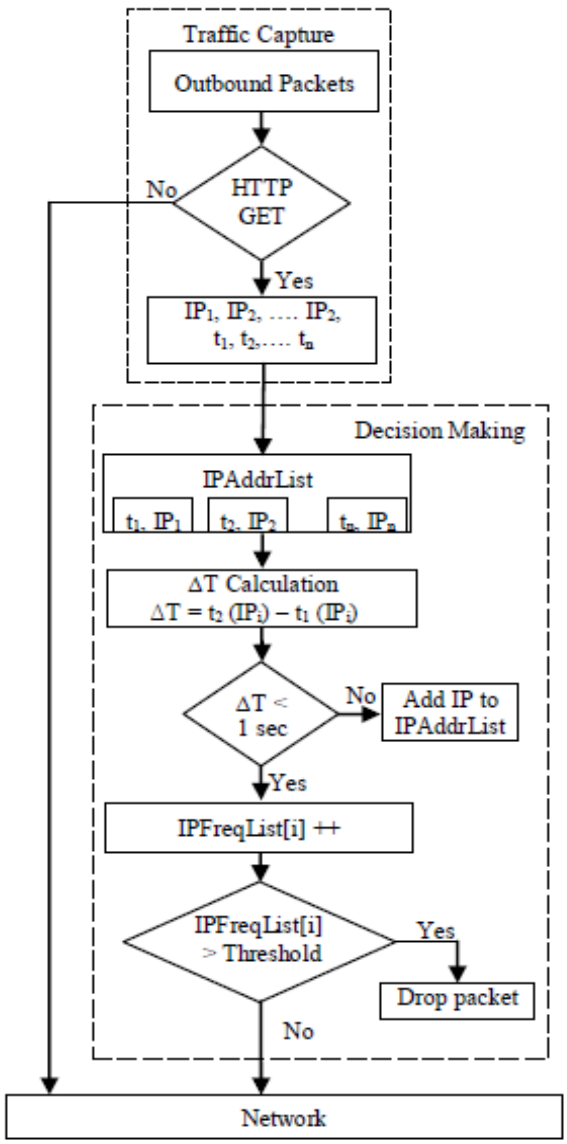


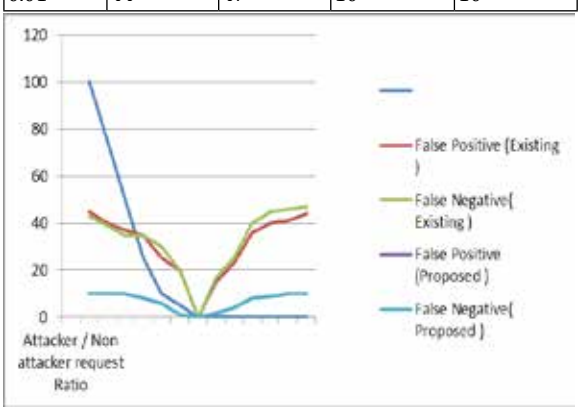
Fig. 2 Flow Chart of TBHF

IV. RESULT AND ANALYSIS

For analyzing the performance of the system some random tests are conducted by providing a combination of attacking and non-attacking request pattern. And compare the results of proposed system with the existing system. That shows a better improvement on the false positive ratio and false negative ratio in my project.

Performance Diagram:

Attacker / non-attacker request ratio	False positive (existing)	False negative (existing)	False positive (proposed)	False negative (proposed)
100	45	43	10	10
75	40	39	10	10
50	37	35	10	10
25	35	35	8	8
10	25	30	6	6
5	20	20	1	1
0	0	0	0	0
0.2	15	17	2	2
0.1	23	25	4	4
0.04	36	40	8	8
0.02	40	45	9	9
0.0133	41	46	10	10
0.01	44	47	10	10



V. CONCLUSION AND FUTURE WORK

Application layer attack has become a major threat to the internet in today's world. The focus of this project was to come out with an effective solution for the detection and prevention of clients from inadvertently taking part in such attacks. Accordingly, a Threshold Based Attack Detection (TBAD) was proposed and implemented in Windows OS using J2SDK. Experiments were conducted by generating HTTP GET attacks and using TBAD for its mitigation. It was evident that the TBAD suppressed the flooding packets and thus prevented the client system from taking part in such an attack.

Besides HTTP flood attacks, this web application level implementation can provide an opportunity to slow down the Brute Force Attacks and Web Vulnerability Scanners. When the detected IP addresses shared with other security components, this also would provide an opportunity to block attackers' access to the web application.

ACKNOWLEDGMENT

The authors gratefully acknowledge the support and facilities provided by Department of CSE, Vedavyasa Institute of Technology. Authors also extend their thanks to the Head of the Department Mrs. S. Kavitha Murugesan for her immense help during the course of the project.

REFERENCE

[1] P. Denning, "The Locality Principle," Comm. ACM, vol. 48, no. 7, pp. 19-24, 2005. | [2] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira, "Characterizing Reference Locality in the WWW," Proc. Fourth Int'l Conf. Parallel and Distributed Information Systems, pp. 92-103, 1996. | [3] S. Yu, "Hidden Semi-Markov Models," Artificial Intelligence, vol. 174, no. 2, pp. 215-243, 2010. | [4] "Handling The Client Request: Http Request Headers" Personally developed and taught by Marty Hall |