A Semantic Rule-based Detection Scheme against Flooding Attacks on Cloud Environment

Chu-Hsing Lin[1], Chen-Yu Lee[2], Shin-Pin Lai[1] and Wei-Shen Lai[3]
[1]Department of Computer Science, Tunghai University, Taichung, Taiwan,
[2]Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan
[3]Department of Information Management,
Chienkuo Technology University, Taichung, Taiwan
{chlin,g99350002}@go.thu.edu.tw, chenyu@cs.nctu.edu.tw, weishenlai@gmail.com

Abstract

With the progress the Internet, more and more applications provide Web services. The presentation of web page has evolved to be dynamic. You also can interact with the web page. Some malicious users have malicious browsing behaviors, such as flooding attack, to waste the resources and bandwidth of the host for web page. Nowadays, more and more web services are developed on cloud computing. Flooding attack on the application layer has no ability to cause denial of service to a Web server on cloud computing. But resources on cloud mean cost. Any waste of resource will cause unnecessary cost. Therefore, in this paper we analyze PHP dynamic pages. According to analysis, we propose a method based on semantic concept to formulate rules to indentify malicious browsing behaviors in order to slice the cost.

Keywords: *Clouds, Flooding attack, Web service, Semantic web*

1. Introduction

Dynamic web page servers face all kind of users and browsing behaviors and among these browsing behaviors some are malicious, such as flooding attack [1, 2]. The flooding attack is easy to detect on the network layer but is harder to detect when occurring on the application layer. Flooding attack on the application layer usually makes normal connection with web page server, and then through browsing behaviors, it wastes the resources of web page, such as CPU time, memory, and bandwidth. It is not easy to cause total denial of service on web page servers built on the cloud by way of flooding attack on the application layer, since this time the attackers are faced with high computation power and high bandwidth web page servers as following development of cloud computing technology [3, 4]. But continuous waste of web page server resources due to attackers will become unnecessary cost that cannot be overlooked by enterprise.

In this paper, we grouped the attacks into five types: group, forward sequence, backward sequence, login, and refresh. Further we proposed three algorithms and semantic policies to identify malicious users by figuring out some characteristics of malicious users and to evaluate the cost of each critical page with the thresholds to determine whether the attacks occur in real time.

2. Threats on the Cloud Web Service

Many web sites have moved to cloud platforms as following development of Cloud Computing [5, 6, 7, 8]. Web sites built on the cloud platform still suffer from HTTP Flooding attack, which will not break down the web page system completely but will still cause waste and extra cost on the system resources.

2.1 Cloud computing environment: We mainly focus on web page service on the cloud platform, of which web servers and application servers are constructed via the virtualization method.

2.2 Attack model: A web site breaks down under attacks such as HTTP Flooding attack. It is due to the facts of resource limitations of computing power and processing ability of the host servers. We grouped all the threats into three types:

a) Computation: attackers cause the system to perform substantial operations by sending specific request packets to increase the computation cost of servers.

b) Communication: attackers download large files repetitively from the web server to waste its network bandwidth.

c) Security: many threats, such as SQL Injection, XSS, and password guessing attacks belong to this type.

3. Methodology

Here we define grammar for the web page: G = {*V*, *T*, *S*, *P*}, where V={*S*}, *S* representing the start symbol, *T*={*a*, *b*, *c*, *d*, *e*, *f*, *g*, *h*, *i*, *k*} represents the set of terminal symbols. *P* is the set of production rules and *P* is expressed as Table 1.

We define deterministic finite automatons (DFA) to describe six malicious characteristics:

A. Forward Sequence: Malicious user browse web page from first page to last page step by step as table 1.

B. Backward Sequence: Malicious user browse web page from last page to first page step by step as table 1.

C. Login: Malicious user try to login in and guess the password illegally as table 1.

D. Back and Forth: Malicious users browse only two web pages and change web page continuously in a short time as table 1.

E. Refresh: Malicious user browses only one page many times in a short time as table 1.

Computation: Malicious user browses any web pages randomly but changes page quickly as Section 4.

Table 1. Parametation for Characteristics by Grammar

Characteristic	Grammar
Forward Sequence	abcdefghijk
Backward Sequence	kjihgfedcba
Login	kkkkk
Back and Forth	Only two page (Ex:abababababab)
Refresh	Only one page (Ex:aaaaaaaaaaaa)
Computation	Random page

When a user connects to the web site, the system analyzes the browsing behavior from user. If user changes page under limit time, the system record the pages in a database. When a user changes page rapidly up within default times, the system derives the automaton to analyze the browsing behaviors of the user. The browsing behavior of user is an input. The automaton will identify the user that is malicious user or not.

4. Computation Algorithm

When the browsing behaviors of the user don't match Forward Sequence, Backward Sequence, Login, Back and Forth, Refresh, we compute the total score of pages that browsing by user.

We use algorithm Initial Setup to define malicious threshold and score of every page. We will determine the max request time T_{max} of pages and set its score S_{Tmax} as 10. Then we let the calendar page be standard. Its score supposes 10.Scores $S[i]$ of other pages refer the standard. Scores of every page are 0.86, 0.65,…, 10, 0.65, 0.2. Finally, we compute the total score S_{Total}.

Table 2. Expresstion for Every Parameter

Parameter	Expression
$R_{st}[]$	Static request time of every page
Tmax	The max static request time of all pages
S_{Tmax}	Score of *Tmax*
$S[]$	Score of every page
S_{Total}	Total score of all pages
$S'[]$	The score after first update
Raverage	Average request time of one page for 24 hrs
$R_{dt}[i][]$	Request time of every page by user
$S''[]$	The score after second update

```
algorithm Initial_Setup ()
 begin
  $T_{max} = Max(R_{st}[i]),\ i = 0..n-1;$
  $S_{Tmax} = 10$
  $S[i] = (R_{st}[i] / T_{max})\ S_{Tmax},\ i = 0..n-1;$
  $S_{Total} = \sum_{i=0}^{n-1} S[i],\quad n = size\quad of\quad site;$
 end.
```

If user changes pages under limit time, we record the page of request by user and compute the score and compare with S_{Total}. If the score of user is over the S_{Total}, we can identify the user that is malicious.

Nowadays, there are more and more dynamic websites on the internet. We use algorithm Update1 to update the malicious threshold and score of every page to detect SQL injection. The SQL request times from user are dynamic. We record the dynamic request time from userand calculate the total request time Total, average dynamic request time $R_{average}$, and $S'[i]$. Now, we suppose the $R_{average}$ is 0.18 sec and the request time of every page. Scores of every page are 1.11, 0.83,…, 0.11. Finally, we compute the total score S_{Total}.

```
algorithm Update1 ()
 begin
  For $i$ =0 to $n$-1
   For j = $i$ to $n$
    $Total = Total + R_{dt}[i][j];$
    $Number{+}{+};$
  $R_{average} = Total / Number;$
  For $i$ =0 to $n$-1
  $S'[i] = \frac{R_{st}[i]}{R_{average}} \times S_{T\max};$
  $S_{Total} = \sum_{i=0}^{n-1} S'[i],\quad n = size\quad of\quad site;$
 end.
```

Every user is interested in different pages. We use algorithm Update2 to update malicious threshold and score of every page to avoid illegal user sending request against the page of lower request time. First, we record the page from requesting by user. Then we calculate the total request times $R_{totaltimes}$ and percent of every page $P[i]$. Finally, we update the scores $S''[i]$ and total score S_{Total}.

```
algorithm Update2 ()
  begin
    For $i$=0 to $n$-1
      $R_{totaltimes} = R_{totaltimes} + R_{times}[i]$;
    For $i$ =0 to $n$-1
      $P[i] = R_{times}[i] / R_{totaltimes}$;
    For i= 0 to n-1
   $S''[i] = P[i] \times n \times S'[i]$
   $S_{Total} = \sum_{i=0}^{n-1} S''[i], \quad n = size \quad of \quad site;$
  end.
```

After we finish algorithm Update1 and Update2, we use new S_{Total} to be threshold. Then we continually identify user malicious or not.

5. Experiment Results

5.1 Experiment Environment

For the experiment, we analyzed our laboratory website in Department of Computer science, Tunghai University, which is an Apache 2.0 web server with pages programed by PHP 5.0 and MySQL 4.0 database. The browser pages are listed in Table 3. We assume these pages a ~ k like grammar in Table 1.

Table 3. Browsing Page for Ordinary Users

Browser Page	Request Time
news	0.02 Sec
lab_intro	0.015 Sec
teacher_info	0.095 Sec
members_info	0.1 Sec
equIPment_info	0.02 Sec
project_data	0.021 Sec
document	0.025 Sec
course_info	0.015 Sec
calendar	0.23 Sec
links	0.015 Sec
user_login	0.005 Sec

5.2 Scenario for Browsing Web Page

To discuss browsing behaviors of a normal user, we use the example of a graduate student visiting the web page of a laboratory. First, he may be interested in research areas and groups of the laboratory. Having some understanding about the laboratory, he may want to find what expertise the supervising teachers have and what projects the laboratory members are involved in.

Normal users would browse documents stored on the web, and their browsing behaviors are hard to predict because the browsing path depend on their own interests, and are randomly, slowly, and unrepetitively.

Malicious users may use web crawler tools, such as spider, to get web page response messages and keep browsing the pages that the request time of pages is short and change pages quickly and regularly.

Finally, the system could identify spider, probe attack, password guessing, DoS attacks by the proposed five policies effectively.

6. Conclusions

In this paper, we proposed three algorithms and policies to identify malicious users by figuring out some characteristics of malicious users and to evaluate the cost of each critical page with the thresholds to determine whether the attacks occur in real time. For the future work, we planned to figure out more and more characteristics to make our method completed and to verify its efficiency, the detection rate, and the false decision rate, and to compare it with other detection methods.

Acknowledgment

This work was supported in part by Taiwan National Science Council under grant: NSC 99-2221-E-029-039-MY3.

References

[1] H. Y. Suen, W. C. Lau and O. Yue, “Detecting Anomalous WebBrowsing via Diffusion Wavelets”, Proc. IEEE Int’l Conference on Communications (ICC2010), Cape Town, SouthAfrica, **(2010)** May, pp. 1-6.

[2] C. H. Lin, J. C. Liu and C. R. Chen, “Access Log Generator for Analyzing Malicious Website Browsing Behaviors”, 2009 Fifth International Conference on Information Assurance and Security, Xian, China, **(2009)** August, pp. 126 - 129.

[3] Y. Xie and S. Z. Yu, “Monitoring the Application-Layer DDoSAttacksfor Popular Websites”, IEEE/ACM Transactions on Networking, vol.17, no. 1, **(2009)** February, pp.15-25.

[4] I. W. Kim and K. H. Lee, “A Model-Driven Approach for DescribingSemantic Web Services: From UML to OWL-S”, IEEE Transactionson Systems, Man, and Cybernetics, Part C: Applications and Reviews,vol. 39, no. 6, **(2009)** November, pp.637-646.

[5] K. Xiong and H. Perros, “Service Performance and Analysis in Cloud Computing”, Proc. 2009 World Conference on Services - I, Washington, DC, **(2009)**, pp. 693-700.

[6] M. D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis and A. Vakali, “Cloud Computing: Distributed Internet Computing for IT andScientific Research”, IEEE Internet Computing, vol. 13, no. 5, **(2009)**, pp.10-13.

[7] Z. Zhang and X. Zhang, “Realization of Open Cloud ComputingFederation Based on Mobile Agent”, Proc. 2009 IEEE InternationalConference on Intelligent Computing and Intelligent Systems, vol. 3, **(2009)** November, pp. 642-64.

[8] S. Subashini and V. Kavitha, “A Survey on Security Issues in Service Delivery Models of Cloud Computing”, Journal of Network and Computer Applications, vol. 34,Issue 1, **(2011)** January, pp. 1-11.