Enhancement of Web Sites Security by Utilizing Web Logs Mining

Ching-Nan Lin[1], Chiung-Wei Huang[2,3] ,and Hahn-Ming Lee[2]

Department of Computer Science and Information Engineering[2]
National Taiwan University of Science and Technology
No.43, Sec.4, Keelung Rd., Taipei, Taiwan 106

Department of Electronic Engineering[1]
Chung Yuan Christian University

Department of Electronic Engineering[3]
Ching Yun University

E-mail: hmlee@mail.ntust.edu.tw

Abstract*-The problem of information security on the Web has become a very important issue nowadays. This paper proposes a Web log mining system to enhance the information security on Web servers. The system aims at detecting application-level attacks, e.g., Backdoor or information leak of CGI scripts, etc., that some security tools couldn't detect well so as to avoid the loss or damage of the enterprises or organizations. In the approach, we not only check the Web ServerLogs but also examine the Web Application Logs such that the well-known application-level instrusions can be detected. Then, the density-based clustering algorithm is adopted to mine the abnormal user behaviors on Web usage logs. The extracted information can benefit the System Administrator on detecting the security holes in their Web systems. Moreover, the mined information can help the administrator discovering security related behaviors and take some appropriate refinements or protections. Afterwards, the experiment indicates the workings of our ideas and proposed system.*

Keywords: Web security, Web log mining, density-based clustering, application-level attack, Information Leakage.

1. Introduction

Web security is a very complex issue which covers computer system security, network security, authentication services, message validation, personal privacy issues, and cryptography[1]. According to survey, most future attacks will be at the application-level[2]. Moreover, the Web Application Security Consortium pointed out the Web security threats nowadays include : Authentication, Authorization, Client-side Attacks, Command Execution, Information Disclosure, and Logical Attacks[3]. Table 1 shows different types of application-level security threat. Though many useful detection systems have been proposed[4][5][6], a lot of strategies have to be designed or proposed for the explosively increasing hacking events.

Due to the fast growth of Internet, most of the Web sites provide not only static Web pages but also dynamic Web pages for supporting more convenient and interactive interfaces for users to acquire their interested information. Unfortunately, those dynamic Web pages with CGI scripts, which are often written in ASP, PERL, PHP, JSP and Net.Data, are not secure [5]. Besides, different Web outputs can be obtained by sending different values to the CGI program. Thus hackers might try different values of parameters or even send illegal values to the Web caused error condition occurred for searching the opportunity to illegally enter the Web Server or database system. However, current professionally designed Web sites still suffer from these kind of security problem[2]. Besides, as more people visit the Web sites, huge amounts of Web log data are accumulated. If the System Administrator wants to find out intrusion behaviors through the log data, the Web log Mining becomes essential and important.

Recently, some promising trends has been focused. Locasto et. al,. proposed a collaborative distributed intrusion detection system that enables distributed hosts communicate and monitor abnormal behaviors in a coopertive way[7]. Burnside et. al., proposed an accelerating application-level security protocols based on hardware cryptographic accelerator[8]. In this paper, we propose a Web log mining system to discover useful information in the application-level in order to prevent the information leakages caused by CGI scripts or backdoors of CGI scripts, and some issues mentioned in [3]. Afterwards, the proposed system provides System Administrator useful information to help them refining their application-level CGI scripts and programs or take some other protections to enhance the security of their Web Sites. Also, this kind of data-mining based system can be used for tracking the unknown hacking behaviors on the Web site management.

2. System Architecture

Figure 1 illustrates the architecture of the proposed system. It includes two sub-modules: (a) Batch module(solid line blocks); (b) Online module(broken

Table 1. Different types of application-level security threat[3].

Type	Example
Authentication	Brute Force guess a person’s password / Insufficient Authentication
Authorization	Insufficient Authorization:permits access to sensitive content or functionality that should require increased access restrictions
Client-side Attacks	Content Spoofing:trick a user believing that certain content appearing on a web site is legitimate and not from an external source
Command Execution	Format String Attack/ OS Commanding/ SQL Injection（search for the possibility to run OS or database system commands to intrude the system）
Information Disclosure	Information Leakage:a web site reveals sensitive data, such as error messages, which may aid an attacker in exploiting the system
Logical Attacks	Denial of Service:the intent of preventing a web site from serving normal user activity

Table 2. An example of the Web Application Log. It keeps the Date, Time, Access Methods, Page Path, and Session Number of the access. Due to it has no IP information, it has to integrate with the Web Sever Log for mining.

Web Application Log				
Session ID	Session Information			
Server session = 357276	/ExecMacro/jobseeker/default.d2w/report	20020701030042	357276	Mozilla/4.0 (compatible; MSIE 5.5; Windows 98; Win 9x 4.90; Hotbar 2.0)
	/IBM/www/HTML/ExecMacro/jobseeker/default.d2w/report	GET		
	?SESSION_RN=357276&USER_AGENT=Mozilla%2F4%2E0+(compatible%3B+MSIE+5%2E5%3B+Windows+98%3B+Win+9x+4%2E90%3B+Hotbar+2%2E0)&command=ExecMacro&SESSION_ID=93c5e78297b637&HOST_ NAME=www%2Ehrgini%2Ecom			

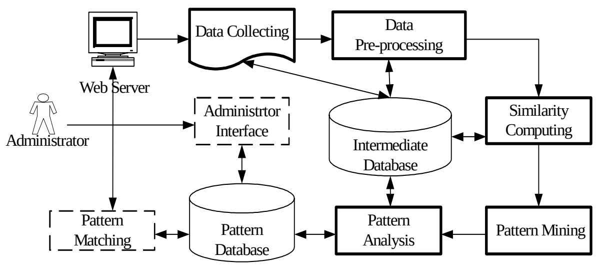

Figure 1. The architecture of the proposed system.

line blocks). The batch module is composed of five task blocks, they are Data Collecting, Data Pre-processing, Similarity Computing, Pattern Mining, and Pattern Analysis. During the operation, the Data Collecting first collects the related Web logs. The Data Pre-processing proceeds the preprocessing task on removing unneeded data. Next, the Similarity Computing aims at calculating the differences on the dedicated logs and the Pattern Mining conducts the finding of abnormal logs. At last, the Pattern Analysis tries to analysis the mining results and store them into the Pattern Database. On the other hand, the online module is designed for online detection of those particular suspicious log patterns which are extracted by the batch module. It has two task blocks, the Administrator Interface and Pattern Matching, shown as the broken line blocks in Figure 1. The Administrator Interface is an interactive interface for the managment or refinement on the Pattern Database. The Pattern Matching block works as an online monitor and checks users’ suspicious behaviors which were mined and found not secure in the Batch mode. In what followings, we detail the function of each task block.

2.1 Data Collecting

The collection of Web log data can be categorized into several different levels. In our proposed system, Web Server Log, e.g., accessing log files, refer log files, agent log files, and Web Application Log files are collected. An access log records all files requests for the Web site. In general, an access log can be analyzed to get the number of visitors, the origin of the visitors, usage patterns in terms of time of day, day of week, and so on. Refer logs indicates the URL that the browser previous visited and it can be used for identifying how people access the site. The agent log details the client programs that were used to access the server. The Web Application Log records application related information, e.g., server session numbers, the URLs of CGI script page, request methods, versions of browsers, etc. Table 2. shows an example of the Web Application Log.

2.2 Data Pre-processing

After the collection of logs, Data Pre-processing is conducted. Figure 2 illustrates the operation of data pre-processing process. It first removes irrelevant fields, e.g., Requests method other than GET and POST, the record that accesses image files (.gif, .jpeg, .swf) or JavaScript files (.js)., the SQL status, SQL syntax of CGI scripts etc., to reduce the mining complexity because the system log file keeps a record of every user's request. Then, the aggregation of these two logs is carried out. Without this kind of cross-reference information on Web Server Log and Application Server Log, Web log mining for application-level attacks would not be able to proceed. Next, the results are saved into an Intermediate Database.

2.3 Similarity Computing

After the log data pre-processed, this section illustrates how to measure the differences among logs such that different behaviors on the Web sites can be distinguished. The definition on the accessing behavior of a user session suggested by Nasraoui **et. al.**[12] is used in our system and is described as follows.

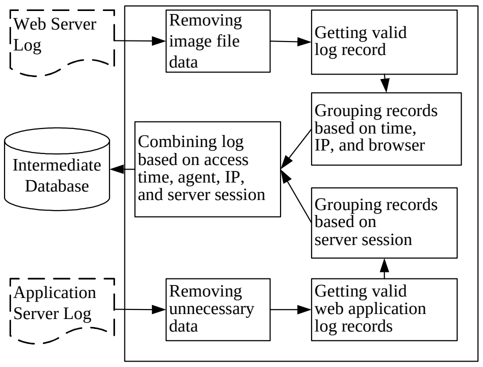
Web Server Log
Removing image file data
Getting valid log record
Grouping records based on time, IP, and browser
Intermediate Database
Combining log based on access time, agent, IP, and server session
Grouping records based on server session
Application Server Log
Removing unnecessary data
Getting valid web application log records

Figure 2. The operation of Data Pre-processing on Web server logs and Web application logs.

Each URL in the Web site is assigned to a unique number $j \in \{1,\ldots,Nu\}$, where *Nu* is the total numbers of valid URLs in the Web site. Thus, the i^{th} server session will be encoded as an *Nu*-dimensional binary vector $p^{(i)}$ with the property:

$$p_j^{(i)} = \begin{cases} 1 & \text{if user accessed j}^{\text{th}} \text{ URL during i}^{\text{th}} \text{ session} \\ 0 & \text{otherwise} \end{cases}$$

$$j \in 1\ldots Nu$$

The first measure between $p^{(k)}$ and $p^{(l)}$ is given by

$$S_{1,km} = \frac{\sum_{j=1}^{Nu} p_j^{(k)} p_j^{(m)}}{\sqrt{\sum_{j=1}^{Nu} p_j^{(k)}}\sqrt{\sum_{j=1}^{Nu} p_j^{(m)}}} \quad \ldots\ldots\ldots(1)$$

where

$p_j^{(k)}$: user accessed jth URL during kth session,

$p_j^{(m)}$: user accessed jth URL during mth session,

$S_{1,km}$: the browsing similarity between kth session and mth session.

Through this measure, we gets high similarity if users browse the same pages during their visit. For example, if click stream of server session A is {/ExecMacro/jobseeker/default.d2w/report, /ExecMacro/jobseeker/resume01.d2w/report} and click stream of server session B is {/ExecMacro/jobseeker/resume01.d2w/report, /ExecMacro/jobseeker/default.d2w/report}, this will receive 1 similarity score according to $S_{1,AB}$. On the other hand, the browsing sequence will be checked in the second step. The second measure of sequence on browsing behavior between server session $s^{(k)}$ and $s^{(m)}$ is formulated as follows,

$$S_{2,km} = \frac{num\_of\_same\_Sequence(s^{(k)}, s^{(m)})}{\sqrt{\sum_{i=1}^{Nu} p_j^{(k)}}\sqrt{\sum_{i=1}^{Nu} p_j^{(m)}}} \quad ..(2)$$

where

num_of_same_Sequence(): the numbers of the same browsing URL sequence between $s^{(k)}$ and $s^{(l)}$,

$s^{(k)}$: the kth server session,

$s^{(m)}$: the mth server session,

$S_{2,km}$: the sequence similarity between the kth session and mth session.

On the same example given above, the similarity between server session A and server session B will receive 0 score according to $S_{2,AB}$ because they didn't have the same sequence during browsing. By using the above two measures, the average of those two similarities can be obtained as follows,

$$S_{km} = Avg(S_{1,km}, S_{2,km}) \quad \ldots\ldots(3)$$

where Avg() is the average function.

At last, the final value of similarity gets 0.5 for server session A and server session B. Figure 3 illustrates the working flow of similarity measure in our system. What follows is an example to illustrate the overall ideas of similarity measure. If there are 3 sessions, A1, A2, and A3, and their browsing information is:

Session A1 : {url1,url2,url3}
Session A2 : {url2,url1}
Session A3 : {url1,url3}

Then we got,

$S_{1,12}$ = 0.816, $S_{1,13}$ = 0.816,
$S_{2,12}$ = 0 , $S_{2,13}$ = 0.41,
S_{12} = (0.816+0)/2 =0.408,
S_{13} = (0.816+0.41)/2 = 0.613.

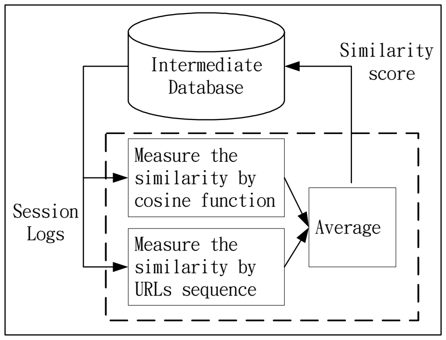
Intermediate Database
Similarity score
Session Logs
Measure the similarity by cosine function
Measure the similarity by URLs sequence
Average

Figure 3. The Similarity Computing on session logs.

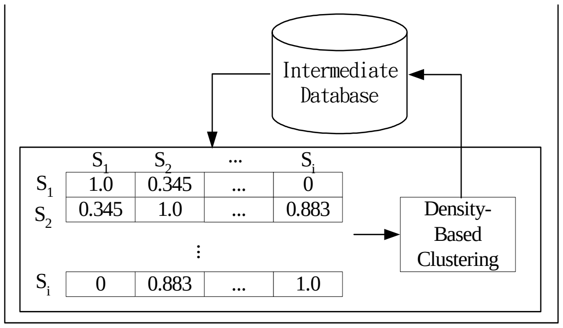

Figure 4. The operation of Pattern Mining.

2.4 Pattern Mining

After the similarities of pair sessions have been calculated, Pattern Mining helps to cluster those browsing behaviors. Due to the behaviors of hackers are quite different from that of ordinary users, i.e., the data mining might face the kind of noisy data problem. And we really don't know how many types of user behavior would be recognized, so applying some clustering methods that set the number of cluster in advance is not feasible.

The density-based clustering proposed by Ester, Kriegel, and Xu [9] has very good characteristics, e.g., (1) it performs well on noisy clustering problem; (2) the number of cluster doesn't need to be set in advance; and (3) the operation is simple but robust. It really meets the requirement of clustering Web logs very well The key idea of density-based clustering is that: for each cluster, the neighborhood area of each clustering within a given radius should cover a predefined minimum clustering members on the same clustering. That is, the density of the neighborhood of each cluster has to exceed some threshold value. Figure 4 shows the operation of using density-based clustering on Pattern Mining in our system.

2.5 Pattern Analysis

Pattern Analysis examines the clustering results of Pattern Mining. Two kinds of analysis are needed to proceed: Normal Cluster Analysis and Noises Data Analysis. Due to the hehaviors of hackers vary a lot, Administrator involved inspection on the Noise Data Analysis is suggested and expected. In what follows, we discuss the detail of each.

(1) Normal Cluster Analysis: When a browsing behavior in server sessions with only Web page visiting, it is easy to be accumulted as a group. We treat those groups as the Normal Clusters. Since less security issues are expecteded to be detected from them, we mainly take the statistics on the parameter value of CGI programs. In addition, these results can help the System Administrator to get more information on the types of parameters for discovery on the analysis rules during users' browsing.

(2) Noise Data Analysis: Those logs on Server sessions, which don't belong to normal clusters, are treated as noises. In general, we are interested in this kind of information. Not only because the amount of it is much smaller than that of the normal clusters but also because these special behaviors get very high possibility on security issues. Of course, there might be some normal activity data tend to be noisy, e.g., same requests over a short span.

Due to its uncertainty, human effort mainly guides this stage currently. In future work, we intend to develop a semi-automatic approach on this. Besides, the checks of parameters' value are conducted in this phase for finding Information Disclosure attacks. Afterwards, the System Administrator can also check the security issues from these patterns. If the security issues are caused by CGI scripting error, they can modify scripts immediately to avoid any kind of loss or damage on the Web site. However if those issues can't be dealt with by any refinements in a short period of time, the Pattern Matching of online module can be used to inform the System Administrators while this kind of the situations occurs again.

2.6 Administrator Interface

This phase belongs to online module. It provides an interface for System Adminstrator to control, manage, and maintain the online monitoring system. In general, special or suspicious patterns of user behaviors can be added or removed by this interface such that Pattern Matching phase can monitor those cases and send an alarm information to the System Administrator to deal with the situation.

2.7 Pattern Matching

Figure 5 illustrates the operation flow of Pattern Matching. It first compares user's requests with stored pattern templates and then redirect requests to Web Application Server for executing CGI scripts if no suspicious cases occur. If not, Pattern Matching will respond an error page to user and record related information in the database. Also, an alarm information is send to the System Administrator for reminding the cases. Thus we protect the Web site to avoid second invasion to fulfil the enhancement of Web site security.

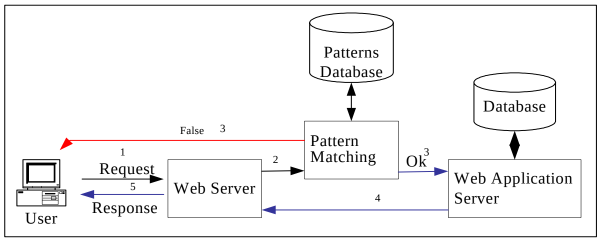

Figure 5. The operation flow of Pattern Matching.

3. System Implementation

In order to conduct experiment by using real Web logs, we asked for the cooperation with a Job seeking Web site, Hrgini [10], in Taiwan. They agreed with our request to collect their Web related logs and analyze their Web site activities on information security. In the experiment, we use the Web log data and Web application log data provided by Hrgini (the cooperative data collecting period is from 2002/07/01 to 2002/07/31). Tables 3 shows the statistics on Web log data and Web application log data. They are the total lines of log, numbers of user sessions, numbers of server sessions, and the numbers of user click stream in each log file individually. Based on the above evaluation on Hrgini, we find that most users just browse the Web site of Hrgini and stay for only a short period of time. However, only a few users stay at the Web site for a long time.

3.1 Result Analysis

After Data Pre-processing, Similarity Computing, and Pattern Mining, some mining results are obtained. Table 4 shows an example of the clustering results on the users' browsing with 3 clicked pages. It has 1341 normal behaviors and 49 noise one. Most of the normal behaviors are browsing the Web pages listed in the right column of Table 4. All of these popular pages are the most representative and informative pages of the Web site. It may said that most people visiting their Web site for just browsing those 3 pages and then going away. In the behaviors with 4 clicked pages, we have 3 clusters with totally 894 normal behaviors and 32 be treated as noises. In 5 to 8 clicked pages, we have 5 clusters with totally 1847 normal browsings and 80 noises. For the space concern, we would not list them here. After analysis, some abnormal patterns are discovered. For example, in Table 5, the server session 364015 on July 05, in which the value of parameter, Prrfnbr, is not unique. Due to the Prrfnbr is used to identify each registered user in Hrgini, the value of it has to be unique. Then, we trace the CGI script, "resume065.d2w", and find that it directly inserts constant value into the tech_skill table(a table in the database), which is used to record the status of each registered users' technique skill. It causes the consistency problem. Also, it reveals an issue on the quality and requirement of the Web site programming. May be, there might cause by the maintenance since the programmer were not in their company anymore. Thus, we suggest that they should modify the CGI script of "resume065.d2w", and it should not use a constant value of Prrfnbr directly in the script. In Table 6, we show the format of the final output of suspicious behaviors through the examination of the noise behaviors.

Table 3. Statistics on Web Server Log and Web Application Log data in Hrgini [2002/07]. (NA denotes Not Available)

Web log Types / Statistics	Web Server Log	Web Application Log
Total lines of log	193216	121678
User Session	37011	NA
Server Session	NA	20628

Table 4. Clustering result on users' browsing with 3 clicked pages.

SN：1390 number of Cluster 1：1341 number of Noises：49	Cluster 1： /jobxxx/default.d2w/report /jobxxx/idv.d2w/report /jobxxx/job1.d2w/sel_job /jobxxx/resume.d2w/report /taxxx/talent.d2w/report

Table 5. An example of abnormal pattern detected on the Server Session 364015. The value of the Prrfnbr is not unique.

Server Session	364015
Click stream length	51
Total amount of parameters	99
URL	/ExecMacro/jobseeker/resume065.d2w /IT2002-07-05-10.00.47

Parameter	Prrfnbr = 5753 (correct)
	Prrfnbr = 1509 (error)

4. Conclusion and future work

In this paper, we proposed a system to discover users' abnormal behaviors, e.g., information leakages in CGI scripts, etc, which might cause application-level security problems and most general security tools couldn't detect them well. The integration on the Web Server Log and Web Application Log is one of our original ideas for extracting more valuable and important patterns that could not be detected by checking only the Web logs. That is we intend to find out security problems caused by Web application programs and suspicious user behaviors. Besides, the appropriate similarity measure and robust clustering method adopted help to find the security related logs. Moreover, we provide the extracted information for System Administrators to appropriately modify their Web application CGI programs to enhance their Web site security. Also, the experiments conducted on a job-seeking Web site in Taiwan indicate the workings of our ideas and the proposed system. In what follows, we point out some future work:

- Continue to test on scalable and long term Web logs for mining more informative information.
- Carefully deal with the parameter selection in clustering because the parameter also affects clustering results.
- Establish a semi-automatic approach for the Pattern Mining

5. References

[1] W3C Security Resources, http://www.w3.org/Security/

[2] David Scott and Richard Sharp, "Developing Secure Web Applications," *Internet Computing*, pp. 226-231, November/December 2002.

[3] Web Application Security Consortium: Threat Classification, http://www.webappsec.org/

[4] A. D. Rubin and D. E. Geer, "A Survey of Web Security," *IEEE Computer*, Vol. 31, No. 9, pp. 34-41, September 1998.

[5] B. Arbaugh, "Security: Technical, Social, and Legal Challenges," *IEEE Computer*, Vol. 35, No 2, pp. 109-111, February 2002.

[6] Bingyang Zhou, "An Integrated Web Security System," Proceedings of the 14th International Workshop on Database and Expert Systems Applications (DEXA' 03), pp. 204-208, September 2003.

[7] Michael E. Locasto, Janak J. Parekh, Sal Stolfo, Angelos D. Keromytis, Tal Malkin, and Vishal Misra. *Columbia University Computer Science Department Technical Report CUCS-012-04,* March 2004.

[8] M. Burnside, A. Keromytis, *Accelerating Application-Level Security Protocols* Proceedings of the 11th IEEE International Conference on Networks (ICON), September/October 2003.

[9] M. Ester, H. P. Kriegel, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," *Proc. of the second International Conference on Knowledge Discovery and Data Mining (KDD'96),* AAAI Press, Menlo Park, CA, pp. 226-231, August 1996.

[10] HRGini Jobseeker Inc., http://www.hrgini.com/

[11] Anupam Joshi, Raghu Krishnapuram, "On Mining Web Access Logs," ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery 2000, pp. 63-69, 2000.

[12] O. Nasraoui, H. Frigui, A. Joshi, R. Krishnapuram, "Mining Web Access Logs Using Relational Competitive Fuzzy Clustering," Proc. of the Eighth International Fuzzy Systems Association World Congress, August 1999.

Table 6. Examples of Suspicious lists. xxx, yyy, and zzz denote the IP addresses.

No.	IP	User ID	Session numbers	Possible threat type
1	xxx.xxx.xxx.xxx	---	363121	Authentication
2	yyy.yyy.yyy.yyy	tom	364015	Information Disclosure
3	zzz.zzz.zzz.zzz	---	364515	Command Execution
…	…	…	…	…
n	qqq.qqq.qqq.qqq	---	365723	Logical Attacks