Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Technology 4 (2012) 500 – 505

Procedia Technology

C3IT-2012

Enhanced Query based Layered Approach towards detection and prevention of Web Attacks

Gaurav Kumar Tak[a], Gaurav Ojha[b]

[a] School of Computer Science & Information Technology, Lovely Professional University, Phagwara, Punjab - 144402, India

[b]Department of Information Technology, Indian Institute of Information Technology, Gwalior, Madhya Pradesh – 474010, India

Abstract

The Internet can be defined as a global system of interconnected networks (wired/wireless) that use a Standard Internet Protocol Suite (Transmission Control Protocol/IP) to serve information worldwide. The client server architecture defines the way in which computing devices all over the world connect to the World Wide Web. In this architecture, the client requests some information from a web server through a web browser. The web server connects to a database server in turn to fetch data. The connection between the web server and the database is the one that needs to be well secured. This is where the role of secure authentication techniques comes into picture.

Cyber-crimes are immoral actions that include illegal access of data, illegal interception of data, eavesdropping of unauthorized data over an information technology infrastructure, etc. There are various kinds of cyber-crimes such as Web attacks, Spam, Phishing Attacks, Information Warfare, Nigerian Scams, and DOS Attacks. At some or the other stage, most of these are ramifications of web attacks – an advanced prevention technique of which is explained in this paper. The proposed methodology utilizes a multi-tier mechanism to detect SQL attacks while maintaining the speed and user experience of the web application. The layered approach ensures that a genuine user would never feel that such a security mechanism was in place, while making it extremely difficult for intruders to break in.

Keywords: SQL Injection, Denial-of-service, Cross side scripting, Knowledge Base, Active Mode, Passive Mode, Brute Force.

1. Introduction

Nowadays, E-commerce, social and business networking sites are experiencing a boom courtesy of the Internet. These applications are expected to behave reliably as they involve enormous transactions of valuable/personal data. The working of these applications is based on web scripting languages such as PHP, JSP, ASP.NET, Perl and the likes in order to format the image and text embedded into the web page [1]. Confidentiality, Availability, Integrity and Non-repudiation (CAIN) are the four salient features of a secure system [2]. SQL, the Structured Query Language, is a language aimed at nurturing data in databases. However, its statements are prone to potential attacks. Thus, it becomes important to develop

doi:10.1016/j.protcy.2012.05.080

techniques to detect and prevent attacks on SQL databases while letting web applications be as user-friendly as possible. The technique discussed in this paper can detect and prevent a wide range of web attacks and works in a way so that a legitimate user shall never be bothered about it.

2. Web Attacks

The most prominent kinds of web attacks are listed below:

• SQL Injection	• Cross Side Scripting	• Remote Command Execution	• Path Traversal

An efficient system should be able to detect and prevent all of these attacks so that the user's data can be kept completely safe and online transactions can take place securely.

2.1. SQL Injection

An SQL injection attack takes place when a hacker changes the semantic or syntactic logic of a SQL text string by inserting SQL keywords or special symbols within the original SQL command that will be executed at the database layer of an application [3]. An SQL injection may consist of simple strings such as {OR 'x'='x';}, or a compound statement aimed at modifying the database records such as {'\"; DROP TABLE profile; --';"}. In the absence of any validation techniques for the input string in forms, SQL injections can be fatal and may result in significant losses of data as well as money.

2.2. Cross Side Scripting (XSS), Remote Command Execution and Path Traversal Attacks

In XSS, malicious data is injected into a database so as to gain unauthorized access to a network connection of an authorized user. Such attacks can occur when data sent to the server are put onto the web site without being properly analyzed for possible security threats. Using Remote command execution attacks, attackers are able to pass some commands to other applications. With Client-Server architecture, the attacker can easily gain admin level privileges, allowing attacks from remote locations on the servers, and can easily execute whatever commands the attacker wishes to perform and also the desired operations, on the web server. Path traversal attacks are performed using some security design errors and not necessarily coding errors. These vulnerabilities facilitate the attacker to access files, directories, (which reside at the web server) and commands for which users are not authorized.

A Knowledge Base is the modeling of previously occurred events in order to predict future events by employing some artificial intelligence techniques [4]. It is a sort of database for knowledge management, providing the means for the computerized collection, organization, and retrieval of knowledge. The basic advantages offered by such system are documentation of knowledge, intelligent decision support, self-learning, reasoning and explanation [5]. In the employed system, a highly simplified Knowledge Base Architecture of artificial intelligence is used.

3. Related Work

Most of the research being carried out, nowadays, pertaining to detection or prevention of SQL attacks can be, in general, divided into three categories (1) Runtime HTTP requests, (2) Design-time web application source code, and (3) Runtime dynamically generated SQL statements. A common artificial

neural network based approach that is used to characterize network traffic is using data-mining techniques. For example, in [6], the authors have described the clustering techniques over unlabeled network traces to detect intrusion patterns. Some of the statistical techniques have also been used to model the network worms' behavior [7]. A longer approach has been used in [8] using parse-tree based validation techniques. Detection approaches based on artificial neural network primarily rely on some features of specific applications and protocols employed by them. For example, in order to recognize sequences of normal system calls for any application, sequence analysis is applied with system calls generated by specific applications [9] [10]. These profiles, specific to applications, are then used to recognize attacks that generate previously undetected sequences.

4. Proposed Methodology

In this paper, we are proposing an enhanced query based layered approach, for detecting SQL attacks and other web attacks, which is built from the ground up, keeping in mind the various intricacies involved in such kind of attacks. Our system uses some knowledge base and query generation using the history of previous attacks and some java script feature which is determined by the user access level. Using the knowledge base, detection of SQL attacks can be easily performed. It also maintains a list of some keywords, which make it easier to detect a large number of attacks at a faster pace.
The steps followed by in the proposed detection approach to identify the SQL injections are as follows:

4.1 *Initial Attack-Validation*: It is the fastest step of the proposed methodology which is based on the historical attack analysis. Algorithm for the attack detection:

```
1) Get input string; 2) Match Input string in attack table; 3) if result=true
Declare "Attack" else Exit ();
```

The above algorithm validates the input SQL string using the ***Initial Knowledge Base*** which stores all the frequent SQL attacks of each category and is managed by the probabilistic approach. If the new input string pattern matches with the any of the patterns already stored in initial knowledge base, then it is declared as an SQL attack and a warning message will be generated automatically.

4.2 *AND-OR Validation*: This step determines the AND-OR word in the input String of web form(s). This Step uses some string comparing operations and parsing techniques to find the AND-OR locations and also counts AND & OR keywords separately before performing the SQL execution (***mysql_query***) operation. The algorithm for this step is given below:

```
1) Get input string; 2) Explode each word; 3) Repeat step 4 and 5 until string ends;
4) Match each word with AND & OR keyword;
5) If result=true /*Comment condition is true for any word*/ Declare "Attack" else Exit ();
```

This step only contains parsing operations to validate the AND & OR keywords in the final SQL string of the user parameters.

4.3 *Equal Sign ('=') Validation*: This step analyzes the '=' syntax in the input String of web form(s) using KMP algorithm [11]. This Step compares the final SQL string of the user parameters with the standard format of SQL query. Algorithm for this step is:

```
1) Get input string; 2) Repeat step 3 and 4 until string ends.
3) Match each letter with '=' keyword;
4) If result=true /*Comment condition is true for any word*/ Declare "Attack" else Exit ();
```

This operation is also executed before execution of the commit statement of the SQL string (***mysql_query***).

4.4 Attack Keywords Analysis: In this algorithm, each letter of input word is exploded and matched with a specific set of attack letters (:,",' ,#,-- ,~). Algorithm for this step is as follows:

1) Get input string; 2) Explode Each letter; 3) Repeat step 4 and 5 until string ends.
4) Match each letter with specific set of letters (:,",' ,#,-- ,~);
5) If result=true /*Comment condition is true for any word*/ Declare "Attack" else Exit ();

4.5) Cross Side Script Attack Analysis: This step is used for the detection of JavaScript-based attacks. Java script is used for the script based analysis. Many times java script can be used to confuse the web users and many times it is used to collect some confidential information about the access level and other database information also. The algorithm for this analysis is as follows:

1) Get input string; 2) Explode each word; 3) Repeat step 4 and 5 until string ends.
4) Match each word with '<script' keyword;
5) If result=true /*Comment condition is true for any word*/
Declare "Cross Side Script Attack" else Exit ();

This type of security can be included in the page-coding itself at the time of development of pages and their design. It proves to be a very efficient method for preventing a wide range of web attacks.

5. Implementation and analysis

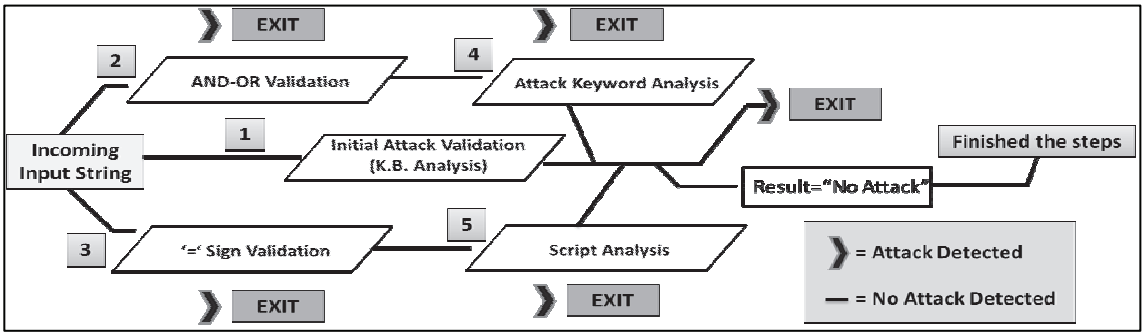

Fig. 1: Representation of proposed methodology

We have implemented the proposed steps in real time scenario, analyzed user inputs and recorded some attacks using above steps and pattern matching techniques. Using the proposed methodology, we are able to detect any misbehavior of users and provide an additional layer on the data security. We have created the proposed environment using HTML, script languages (PHP), AJAX, XML, Apache Server and MySQL. JSP and ASP.net also can be used as script languages, in which case we need Apache Tomcat and Windows Server (IIS) to process and execute code written in JSP and ASP.net respectively.

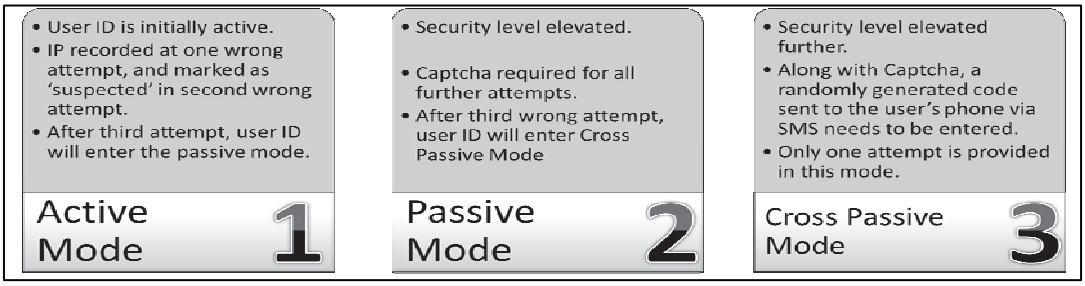

Fig. 2: Representation of layers of security in the web application (During implementation)

Figure 1 represents the different steps in the implemented scenario. Each step is explicitly defined and provides two distinct results using if-else conditions. We have recorded and analyzed each SQL query before its execution and compared the changes in both stages (before and after execution). Figure 2 describes the actual implementation of the methodology in a web application where user activity is restricted using techniques such as CAPTCHA [12] and multi-environment verification codes. The following table data represents the recorded activities over the various types of attacks, their detection using the above steps. There were a total of 1104 attacks out of which 1087 were detected. Efficiency and time complexity of the results depend on the server configuration and pattern of the scripts.

Table 1. Data of recorded activities

	Initial Attack Validation	AND-OR Validation	'=' Sign Validation	Script Analysis	Detected Attacks
Recorded Activities	129	461	393	104	1087
Execution Time (Per Attack)	0.07 seconds	0.2 seconds	0.178 seconds	0.3 seconds	-

6. Conclusion and Future Work

Our work is inspired by a situation of large number of SQL attacks, SQL injections and JavaScript attacks over various web forms. We have recorded all the input strings which are responsible for the query execution and analysed them thoroughly using the described steps. We have executed the methodology on an online testing platform and recorded, analysed all the SQL strings. The experiment results provide complete scenario of the problem and accuracy of proposed steps. Our system indicated that the attacks were detected with 98.46% accuracy. Advantage of the methodology is that it gets executed in lower time and space complexities. In this way, it doesn't affect execution speed of the web page and performance of the server system

References

1. CERT/CC. "Code Red Worm", "Exploiting Buffer Overflow", In IIS Indexing Service DLL. Advisory CA-2001-19, July 2001.
2. Dhiraj, G., Nilkanthrao, RSA Based Confidentiality And Integrity Enhancements in SCOSTA-CL, A thesis report, Department of Computer Science and Engineering, Indian Institute of Technology, Kanpur, India, (July,2009).
3. Halfond, W. and Orso, A., AMNESIA: Analysis and Monitoring for Neutralizing SQL Injection Attacks, 20th IEEE/ACM international Conference on Automated Software Engineering, pp. 174--183. USA, New York, (2005).
4. J. Ullman, Database and knowledge base systems, In Database and knowledge base systems, Volume 2, Computer Science Press, 1989.
5. Akerkar RA and Sajja Priti Srinivas, *"Knowledge-based systems"*, Jones & Bartlett Publishers, Sudbury, MA, USA (2009).

6. L. Portnoy, E. Eskin, and S. Stolfo, Intrusion Detection with Unlabeled Data Using Clustering, Proceedings of ACM CSS Workshop on Data Mining Applied to Security, Philadelphia, PA, November 2001.
7. M. Liljenstam, D. Nicol, V. Berk, and R.Gray, Simulating realistic network worm traffic for worm warning system design and testing, In Proceedings of the ACM Workshop on Rapid Malcode, pages 24–33, Washington, DC, 2003.
8. Buehrer, G., Weide, B. W., and Sivilotti, Using parse tree validation to prevent SQL injection attacks, Proceedings of the 5th International Workshop on Software Engineering and Middleware (Lisbon, Portugal, September 05 - 06, 2005), SEM '05, ACM, New York, NY, P. A. 2005, 106-113. DOI= http://doi.acm.org/10.1145/1108473.1108496.
9. S. Forrest, A Sense of Self for UNIX Processes, Proceedings of the IEEE Symposium on Security and Privacy, pages 120–128, Oakland, CA, May 1996.
10. C. Warrender, S. Forrest, and B.A. Pearlmutter, Detecting intrusions using system calls: Alternative Data Models, IEEE Symposium on Security and Privacy, pages 133–145, 1999.
11. Thomas H. Cormen, Charles E. Leiserson, Ronald L.Rivest and Clifford Stein, Introduction to Algorithms, MIT Press/ McGraw-Hill, 2001.
12. Luis von Ahn, Manuel Blum, Nicholas Hopper, and John Langford, CAPTCHA: Using Hard AI Problems for Security in Eurocrypt.