Advanced Query-based Multi-tier Approach towards Detection and Prevention of Web Attacks

Gaurav Kumar Tak
School of Computer Science and Information Technology
Lovely Professional University,
Phagwara, Punjab - 144402, India

Gaurav Ojha
Department of Information Technology,
Indian Institute of Information Technology and Management,
Gwalior, Madhya Pradesh – 474010, India

ABSTRACT
The Internet, which can be defined as a huge network of networks - both wired and wireless, uses the Internet Protocol Suite (TCP/IP) to make information available beyond geographical boundaries. Computing devices all through the world connect to the World Wide Web via the Client Server architecture. 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.

Nowadays, Cyber-crimes are becoming rampant. These include illegal access of data, illegal interception of data, eavesdropping of unauthorized data over an information technology infrastructure, etc. Popular Web attacks include Spam, Phishing Attacks, Information warfare, Nigerian Scams, and Denial-of-Service attacks. At some or the other stage, most of these are ramifications of web attacks and SQL attacks – practical implementation of an advanced analysis and prevention technique of which is explained in this paper. It uses a multi-tier approach which makes web applications retain their simplicity for the user and complexity for the attacker.

General Terms
Web Attacks, SQL Injection, Cyber Crimes and Cyber Security

Keywords
Denial-of-service attacks, XSS, Brute Force, Dormant Phase, Alert Phase and Inquisitive Phase.

1. INTRODUCTION
A very important part of our day-to-day life, nowadays, is Information Technology. It lies at the heart of almost all advanced technologies that make human life simplified. The number of E-commerce sites, Social Networking sites and other web portals is on the rise. Large amounts of data are stored in the databases of web portals, the working of which is based around web technologies such as PHP, ASP.NET, JSP, XHTML and SQL. A simple web application usually consists of a PHP server (such as Apache), an SQL database and an XHTML front-end which forms the user-interface. Due to the large amount of penetration of web services in almost all spheres of human life, it becomes important to make sure that these services are not tampered with, in any form.

With an increasing number of web portals, there is also an increase in the number of web developers, who either lack technical expertise, or are too much focused on the end-user requirements so as to ignore security issues in the web applications they have developed. Security of a web application should be focused on at the time of development of the application itself. Since most of the web is largely open, it becomes very easy to find security loopholes and turn insecure web applications into potential victims for exploitation using worm attacks, etc. [1].

As per [2], a secure system must have the four features defined by the CAIN architecture namely, Confidentiality, Availability, Integrity and Non-repudiation. Web applications, especially E-commerce and Social Networking sites must fulfill these requirements, in order to be classified as secure applications. But it has been found that most of them don't. Most of the internet applications these days are based on dynamic scripting (PHP/JSP) in order to allow on-demand fetching of data from SQL databases. A database designed for this purpose might contain a lot of confidential information such as user passwords, bank account details, and other personal information. It is considered as a security violation if it is possible to gain unauthorized access to such a database, sneaking into personal data. An attacker might also misuse this data which may be devastating at times. The Structured Query Language (i.e. SQL) is aimed at maintaining and nurturing data is databases. But SQL statements are very prone to harmful modifications which can turn them into attacks sequences. A secure system must be able to compare the data during usual behavior with that during an observation in order to detect an attack. In this paper, the proposed technique can detect a large number of web attacks prevalent over modern web applications.

2. ATTACKS OVER THE WEB
Web attacks also come under the category of cyber-crimes. The popular web attacks during present times are [4]:

- SQL Attacks
- XSS (Cross Site Scripting)
- Attacks over Remote Connection
- Attacks with Code Analysis

2.1 SQL Attacks
An SQL injection attack takes place when a hacker changes the semantic or syntactic logic of an 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 example of an insecure PHP code that can be used for login is given below:

```
$con=mysql_connect("localhost", "root", "gaurav");
$condb=mysql_select_db("admission",$con);
$username=$_REQUEST['username'];
$password=$_REQUEST['password'];
$sqlstring= "Select * from login where userid= '$username' and password = '$password' ";
```

```
$sqlexecution=mysql_query($sqlstring);
if(mysql_num_rows($sqlexecution)==1)
echo "Login Successful";
else
echo "Login Failed";
```

The above code validates the users. The data input by the user are assigned to PHP variables *username* and *password* and then inserted to the SQL query. If the SQL query results in one row, it means the user is authorized to access the database and if it results in no row then the user is not authorized. Since data input by the user are not analyzed, it is possible to inject any combination of strings. For example, an attacker can use: OR 'a'='a' in the username field and let the password field be empty. Now the modified SQL query will be as described below:

```
$sqlstring= "Select * from login where userid= '$username' or 'a'= 'a' and password = " ";
$sqlexecution=mysql_query($sqlstring) or die("Query execution failed");
```

In the above SQL query, the Boolean expression 'a'= 'a' is always true, so it allows the attacker to access the database without entering the correct password in the password field of the web form. Thus, an attacker will be easily logged in with privileges same as those of the other (legitimate) users stored in the login table of the database.

2.2 XSS – Cross Side Scripting

Cross Site Scripting is yet another variety of attacks on Web applications. In this malicious data is injected into a database so as to gain unauthorized access to a network connection of an authorized user. Websites, generally, employ scripts written in JavaScript coupled with HTML, which runs on a client side rendering application for seamless user experience. Attackers utilize the fact that there is a trust relationship between a Web server and a browser. Such attacks can occur when data sent to the server are put onto the web site without being properly analyzed for possible security threats. If the data input in a form is a nasty script, it will be run by the browser. In the simplest case, a user will be shown pop-up window with its *session ID* completely recognizing it.

2.3 Attacks over Remote Connection

Using Remote Connections, attackers are able to pass some commands to other applications. With Server architecture, the attacker can easily gain admin level privileges, thus allowing attacks from various remote locations on the servers and can easily execute whatever commands he wishes to perform and also the desired operations, on the web server.

2.4 Attacks with Code Analysis

These 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 [5]. It is a sort of database for knowledge management, providing the means for the computerized collection, organization, and retrieval of knowledge. Also a collection of data representing related experiences and their results is related to their problems and solutions.

They are basically artificial intelligence tools providing intelligent decisions. Knowledge is obtained and represented using various knowledge representation techniques rules, frames and scripts. The basic advantages offered by such system are documentation of knowledge, intelligent decision support, self-learning, reasoning and explanation. [6]

In the employed system, a highly simplified Knowledge Base architecture of artificial intelligence is used.

3. PREVIOUS 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. In order to detect SQL attacks, some researchers employ only one type of data while some others employ two. We, now, discuss some of the works done in the field of SQL attacks and their preventions based on the above categorization, and give a short summary of the advantages and shortcomings of these methods.

Detection systems which are based on artificial neural network techniques, represents several approaches that build normal profile based on the training data, behavior of users and web applications. A common approach is used to characterize network traffic that is data-mining techniques. For example, in [7], 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 [8].

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. As another example, some authors have described about utilization of the statistical analysis related to the traffic of network to study the normal behavior of applications based on network. This is achieved by analyzing packet header information as well as the contents of application specific protocols.

4. PROPOSED METHODOLOGY

In this paper, an advanced query based multi-tier approach, for detecting SQL attacks and other web attacks, is being proposed, designed from the base, after realizing the complexities involved in these attacks. Knowledge based dynamic query generation techniques have been implemented in the designed application, which learns from the history of previously occurred attacks over the system. The system gains efficiency with time. It also maintains a list of common attacks which helps detect a larger number of attacks at an improved rate.

The proposed methodology comprises of certain steps in order to detect attacks, which are explained in the following article:

4.1 K.B. Cross-check

This is perhaps the fastest step in the proposed methodology. It 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:

```
1) Receive input string;
2) Match Input string with those in attack table;
3) If result=true; Declare "Attack" else
Exit ();
```

In the above algorithm, if the 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. In case the input string is not in the K.B. but is identified as an attack at a later stage, it will be stored in the K.B. for future reference. This step also improves the efficiency of the system with time.

4.2 AND-OR Validation

In this step, the system detects the presence of AND-OR tokens in the input string, using string comparison and parsing operations. The algorithm, which detects and counts the number of AND-OR tokens separately, is given below:

```
1) Receive input string; 2) Split each word;
3) Repeat step 4 and 5 until string ends;
4) Match each word with AND & OR token;
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 tokens in the final SQL string of the user parameters.

4.3 Equal Sign ('=') Validation

In this step, the system looks for the presence of an '=' sign within the input string. Algorithm for this step is as follows:

```
1) Receive input string;
2) Repeat step 3 and 4 until string ends.
3) Match each letter with '=' token;
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 Tokens Analysis

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

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

4.5 Tracing Cross Side Scripts

In this step, the system detects the presence of JavaScript-based attacks using JavaScript itself. Sometimes, attackers use JavaScript to confuse users while at times it is used to extract some confidential information about cookies and access level, etc. Following is the algorithm for tracing cross side scripts:

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

The security discussed here can be included in the page coding itself at the time of development of the web application. The whole methodology has been found to be quite effective in order to prevent practically all kinds of SQL and JavaScript attacks.

5. IMPLEMENTATION OF PROPOSED METHODOLOGY

All through the development of the proposed methodology, it has been observed that user-friendliness is of utmost importance. In order to make sure that the secure web applications are silent for the genuine user at the same time, the following multi-tier approach is being proposed –

5.1 Dormant Phase

User ID is initially in Active Mode. The user needs to enter his username and password. If the username and password matches with those stored in the database of the server, the login is authenticated and the user can use his account. If the username and password do not match, but username exists in the database, the user is given another chance to enter his credentials for login at the time of which his IP and username will be recorded. If the credentials do not match with the stored ones in the second attempt the user will be given another attempt and his user ID will be marked as 'suspected'. If the login is unsuccessful even after the third attempt, the user ID will enter into 'Alert Phase'.

5.2 Alert Phase

In Passive Mode, the security level is elevated. Now user has to enter, in addition to his username and password, an additional Image Captcha. This step validates a human user. A machine or computer script cannot go beyond this step. If the user enters the correct username and password, and also the image Captcha properly, then his login is authenticated and he can use his account. If the password or Captcha do not match for the first time, the user is given another chance to enter his credentials. If the entered credentials do not match with the stored ones the second time, the user will be given one more attempt before marking the username as 'Inquisitive'. If the username or password or Captcha are entered wrongly one more time, the username enters 'Inquisitive Phase'.

5.3 Inquisitive Phase

In Inquisitive Phase, the security level is elevated further. Now, in addition to the username, password and image Captcha, the user needs to enter a special code that will be sent via SMS. This secret code or SMS code will be sent to the phone number which had been provided by the user at the time of registration. The SMS code will be a randomly generated string. If the user enters his username, password, image Captcha and the SMS code correctly, only then login will be authenticated, failing which he won't be able to login into the account.

5.4 Block Open Accesses

All open accesses, such as remote login, common attack ports, should be kept blocked. FTP connection can still be initiated to view files on server. This will prevent any attacks on the database server that happen due to unauthenticated login and remote access.

5.5 Prevention of Brute Force

Alert phase helps to block brute force attacks by not letting the user to enter the wrong password more than thrice. An image captcha is considered safe and cannot be entered by a script or software. In case the attacker manually enters captcha then he has only three more attempts after which an SMS code will be sent to the original user of the account. This completely washes away the possibility of trying out more combinations of strings for the password. Thus, in this methodology, there is a very high level of security and the user can be assured that the chances of his/her account being compromised are extremely low.

We have implemented the proposed steps in real time scenario, analyzed the user inputs and recorded some attacks using above steps and pattern matching techniques. The analysis of attacks also represents the efficiency of the proposed methodology itself. 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 some web technologies, HTML, script languages (PHP), AJAX, XML, Apache Server and MySQL tools for implementing the methodology. 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. We also applied some basic concepts of PHP, AJAX, MySQL and JavaScript from the references [11] [12] [14].

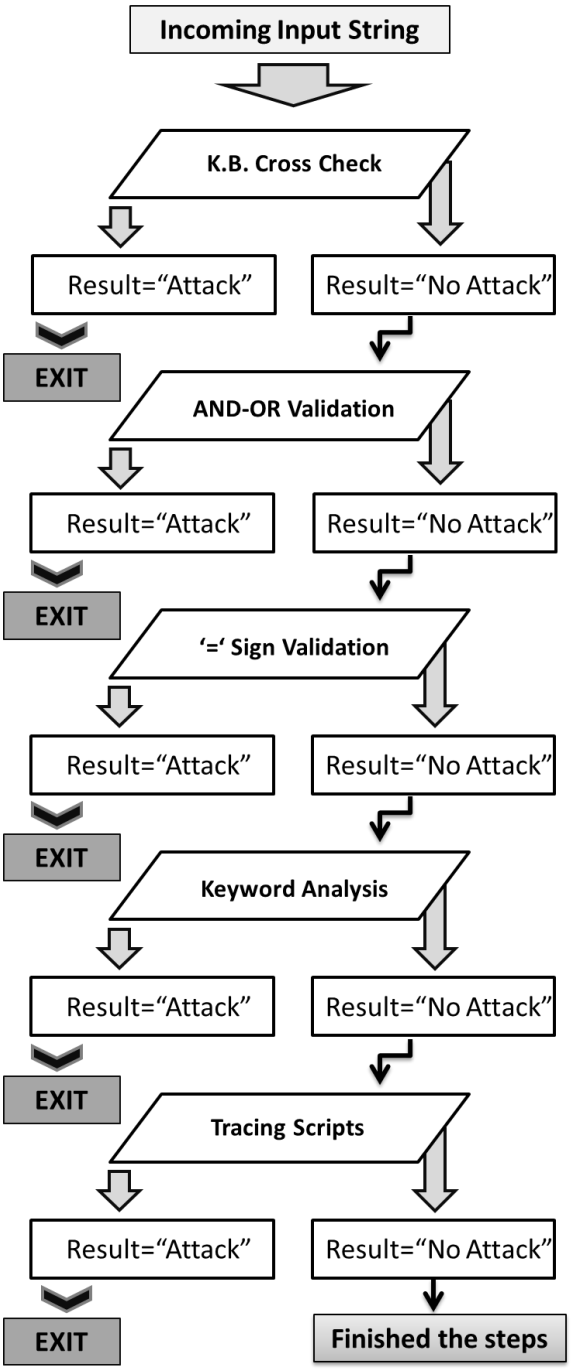

Figure 1: Representation of proposed methodology

Figure 1 represents the different steps in the implemented scenario. Each step is purely defined and provides two types of results using if-else conditions. We have recorded all the incoming input strings over the client–server architecture of a real time system. We have analyzed each SQL query before its execution and always compared the changes between both stages (before and after execution). The following table data represents the recorded activities about the various kinds of attacks and their detection using the above steps.

Table 1: Data of recorded activities

	Recorded Activities	Execution Time (Per attack)
Total Attacks Performed	806	-
Total Attacks Detected	794	-
K.B. Cross-Check	94	0.07 seconds
AND-OR Validation	337	0.193 seconds
'=' Sign Validation	287	0.18 seconds
Script Analysis	76	0.28 seconds

We can get the performance information of the proposed methodology using the experimental results which are shown in Table 1. There were a total of 806 attacks out of which 794 were detected, giving an accuracy of 98.51 % and we can easily compare these results and performance with the previously described approaches of detection of SQL attacks. However, it must be noted that, Efficiency and time complexity of the results depend on the server configuration and pattern of the scripts.

6. CONCLUSION AND FUTURE WORK

In our work, we have recorded all the input strings which are responsible for the query execution and analyzed them using the above described steps. We have executed the methodology on an online examination system and recorded and analyzed all the SQL strings over a significant period of time. These attacks were carried out by some students of universities who provided technical support during analysis part of the methodology. The experimental results provide the complete scenario of the problem and accuracy of above steps. Our system indicated that the attacks were detected with 98.51 % accuracy. Table 1 represents details of all the recorded attacks and detected attacks over the specific time period.

The proposed methodology is advantageous, as it doesn't affect the speed and performance of the web applications due to lower space and time complexities of the detection mechanism. Thus, the whole system is very much secure and user-friendly at the same time.

6. REFERENCES

[1] CERT/CC, July 2001. Code Red Worm, Exploiting Buffer Overflow, in IIS Indexing Service DLL. Advisory CA-2001-19.

[2] Dhiraj, G., Nilkanthrao, July 2009. RSA Based Confidentiality and Integrity Enhancements in SCOSTA-CL, A thesis report, Department of Computer Science and Engineering, Indian Institute of Technology, Kanpur, India.

[3] Halfond, W. and Orso, A., AMNESIA, 2005. Analysis and Monitoring for Neutralizing SQL Injection Attacks, 20th IEEE/ACM International Conference on Automated Software Engineering, pp. 174--183. USA, New York.

[4] http://www.applicure.com/solutions/web-application-security

[5] Ullman, J., 1989. Database and knowledge base systems, In Database and knowledge base systems, Volume 2, Computer Science Press.

[6] Akerkar, R. A., and Srinivas, Priti, Sajja, 2009. Knowledge-based systems, Jones & Bartlett Publishers, Sudbury, MA, USA.

[7] Portnoy, L., Eskin, E., and Stolfo, S., November 2001. Intrusion Detection with Unlabeled Data Using Clustering, Proceedings of ACM CSS Workshop on Data Mining Applied to Security, Philadelphia, PA.

[8] Liljenstam, M., Nicol, D., Berk, V., and Gray, R., 2003. 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.

[9] Forrest, S., May 1996. A Sense of Self for UNIX Processes, Proceedings of the IEEE Symposium on Security and Privacy, pages 120–128, Oakland, CA.

[10] Warrender, C., Forrest, S., and Pearlmutter, B. A., 1999. Detecting intrusions using system calls: Alternative Data Models, IEEE Symposium on Security and Privacy, pages 133–145.

[11] PHP, AJAX, MySQL and JavaScript Tutorials, http://www.w3schools.com/

[12] Von Ahn, Louis, Blum, Manuel, Hopper, Nicholas and Langford, John, CAPTCHA – Using Hard AI Problems for Security, In Eurocrypt.

[13] Cormen, Thomas, H., Leiserson, Charles, E., Rivest Ronald, L., and Stein, Clifford, 2001. Introduction to Algorithms, MIT Press/ McGraw-Hill.

[14] History of PHP and related projects, http://www.php.net/history.

[15] Buehrer, G., Weide, B. W., and Sivilotti, 2005. 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, 106-113. DOI= http://doi.acm.org/10.1145/1108473.1108496