International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Web Filtering with SQL Injection

Yogesh Ghuse[1], Chetan Harshe[2], Pratibha S. Ghode[3]

[1, 2]Computer Engineering Department, Suryodaya College of Engineering & Technology, Nagpur, India

[3]Assistant Professor, Computer Engineering Department, Suryodaya College of Engineering & Technology, Nagpur, India

Abstract: ***As more businesses and organizations provide online services, the number of web sites or applications which are linked to a database has increased greatly. Often the data held in such databases is confidential or private – and possibly of great interest to a hacker, disgruntled employee, or criminal group. While the database and the server holding it may have been secured, the design of the web interface is often overlooked and could allow unauthorized users access to the database. SQL injection, the use of database commands in the SQL language where user input is expected, remains a top threat. It was the 3rd listed error in the January 2009 "CWE/SANS Top 25 Most Dangerous Programming Errors"[a] and has been the mechanism for a number of prevalent attacks. For example, through most of 2008 there were ongoing, indiscriminate and widespread attacks on vulnerable web sites, which added a link to a malicious file (usually JavaScript) that most web site visitors would unintentionally run on loading the page. This then caused the visitor's computer to be infected with malware. Even well-known and widely trusted web sites were affected by this problem. This document will illustrate some of the main techniques used in SQL injection, then describe methods that can reduce the effectiveness of such attacks. In addition to usual standard IT best practice, such as logging and regular and prompt patching, the majority of SQL injection vulnerabilities can be moderated through careful and robust programming. It is hoped that the information provided here will highlight the seriousness of leaving this type of flaw unaddressed and promote the improved design of database-linked Internet resources.***

Keywords: Web Application, SQLIA, Detection, Prevention, Vulnerabilities.

1. Introduction

Structured Query Language (SQL)[b] is used to interrogate and manage relational databases such as Microsoft SQL Server, MySQL, Oracle, PostgreSQL and Sybase. While there is an ANSI/ISO standard for SQL defining elements such as keywords and grammar, most of the major implementations do not employ the full SQL standard and add implementation-specific procedural extensions. For this reason much SQL code is written for a certain target platform and is unlikely to transfer easily to another SQL injection occurs when an SQL instruction is entered in a field of an application or web page that a user can change in an attempt for it to be passed to and executed by the back-end database.

There are a number of forms of SQL injection, which can be broadly separated into input validation circumvention and blind SQL injection. While some attacks are directed at a weakness in the database software, the majority of attacks seen use a flaw in the interface to the database to (without permission) access, add, or modify data, or to execute a command on the server itself. Clearly there can be serious consequences when a malicious SQL injection attack succeeds, affecting the confidentiality, integrity and availability of the data and services it supports. The purpose of this document is to demonstrate why it is necessary to code web pages and applications securely, giving examples of SQL injection attacks and to summaries some ways that these systems can be secured.

The examples discussed relate to web pages or web applications as these are a frequent target of SQL injection attacks and the majority of weaknesses are straightforward to resolve. These are non-invasive illustrations to demonstrate the hazards and principles of SQL injection: a real attacker would have few hesitations in applying techniques that may cause considerable damage. The OWASP Foundation has produced two tools that can be used to learn about and analyze attacks. The application has been developed to demonstrate web application security errors, including SQL injection, and educate developers in how to avoid them. A web proxy, such as OWASP's Web Scarab, is needed to complete some of the activities.

2. Literature Survey

Boyd, Keromytis-2004 proposed SQLr and which uses instruction set randomization of SQL statement to check SQL injection attack. It uses a proxy to a append key to SQL keyword. A de-randomizing proxy then converts the randomized query to proper SQL queries for the database. The key is not known to the attacker, so the code injected by attacker is treated as undefined keywords and expressions which cause runtime exceptions and the query is not sent to database. The disadvantage of this system is its complex configuration and the security of the key. If the key is exposed, attacker can formulate queries for successful attack.

Russell A. McClure and Ingolf H. Kruger-2005 proposed SQL DOM (SQL Domain Object Model): a set of classes that are strongly-typed to a database schema. It is based on compile time checking of dynamic SQL statements. Instead of string manipulation, these classes are used to generate SQL statements. We show how to extract the SQL DOM automatically from an existing database schema, demonstrate its applicability to solve the problems, and evaluate its performance.

Ke Wei et al.-2006 proposed A novel technique to defend against the attacks targeted at stored procedures. This technique combines static application code analysis with runtime validation to eliminate the occurrence of such

attacks. In the static part, we design a stored procedure parser, and for any SQL statement which depends on user inputs, we use this parser to instrument the necessary statements in order to compare the original SQL statement structure to that including user inputs. The deployment of this technique can be automated and used on a need-only basis. We also provide a preliminary evaluation of the results of the technique proposed, as performed on several stored procedures in the SQL Server 2005 database.

3. Exsiting System

Most of existing techniques, such as filtering, information-flow analysis, penetration testing, and defensive coding, can detect and prevent a subset of the vulnerabilities that lead to SQLIAs.

4. Proposed System

SQL injection is one of the main issues in database security. It easily affects the database without the knowledge of the database administrator and the user. It is a technique that may corrupt the information in the database i.e. deletes or changes the full database or records or tables. To exploit the database system, some vulnerable web applications are used by the attackers. These attacks not only make the attacker to breach the security and steal the entire content of the database but also, to make arbitrary changes to both the database schema and the contents. SQL injection attack could not be realized about information compromization until long after the attack has passed in many scenarios, the victims are unaware that their confidential data has been stolen or compromised. SQL Injection attacks can be performed by attackers just with the help of simple web browser. The following section describes the attacks with an example.

Generally the Authenticated users have username and password such as,
Username: Rahul
Password: 123
The SQL Query format will be as follows,
Select * from table where username='rahul' and pwd='123';
The above query then retrieves the needed records from the database where username and pwd is available in the database or it shows some error messages to the browsers. The unauthorized users or the attackers inject the following SQL Injection in this field:
Username: r hul
Password: 123
Then the dynamic SQL query constructed from the above information is,
Select * from table where username=' r hul ' and pwd=123;

In this SQL statement, the actual username is ‘rahul’ which is modified as ' r hul’ by the attackers while generating the Query. This includes the image of ‘a’ in place of the character ‘a’. The attacker will now have the capability of attacking the database by writing the injection code at the inside location of the image (just as-image processing). Here the username is visible as the set of characters so that even by using the character level tainting, it will consider this query as the character and performs the operation. When the image is taken to compare with the string in the Meta Strings library, due to the unavailability of information about the images, the Meta strings library doesn’t consider it as a malicious code and accepts the string. Hence the Query was sent to the database. The result of this query performs SQL Injection attacks.

5. Plan of Action

Table 1: Plan of Action

Month	Plan Of Action
Dec. 2014	Study Of Literature Survey.
Jan. 2015	**Module1** Creating a bookseller website, in which registered user can enter into system and purchase book.
Feb. 2015	**Module2** In which unauthorized user with the help of SQL code and enter into system and hack the system.
Mar. 2015	**Module3** Ceating filters for sql code.
April.2015	Deployment of all modules and Final testing.

6. Problem Definition

A new approach for protecting Web applications-An Image level Tainting, involves comparing the SQL statements that includes the images viewed as characters in the user input with the Meta strings library, to prevent them if found any and protecting the web applications against SQL injection is discussed in this paper. This project includes the strange idea of combining the Indication based method and the Inspection Method. The main problem that occurs with web application security is the SQL Injection, which gives the attackers unauthorized access to the database that contains the Web applications. This leads to the cause of calamities in the Web applications and this is very serious. In Indication based method point of view, it exhibits detection mode for SQL injection using coupled way routing arrangement of amino acid code formulated from web application form parameter sent via the web server. On the other hand from the Inspection based method point of view, it analyzes the transaction to find out the malicious access. In Indication based method it uses an approach called Beschermen algorithm, not only to prevent the SQL Injection attacks, but also reduces the time and space complexity. This system was able to stop all of the successful attacks.

7. Architecture

Our approach against SQLIAs is based on signature based approach that easily addresses the security problems related to input validation. This approach describes two modules which are used to detect the security issues. Comparative module has got the statement from the web application which includes both the Hirschberg Algorithm to analyze the Statement into the set of characters, and Beschermen algorithm to compare the each character with respect to the Reciting module. After the each character in the statement is scrutinized, if it finds any suspicious activity like finding the images in place of characters, it acts as an active agent to stop the transaction and audit the attacks.

Reciting module includes the Meta strings library which comprises the predefined keywords and is updated with new type of information in terms of coordinates of the images,

their pixels information, color resolution and the details on type of images (extension files like .jpg etc). If both comparative module and inspection module has satisfied, it provides the complete transaction. The following figure 1 clearly portrays the architecture of the system to prevent the SQL Injection attacks using this new approach. The following section outlines each module's work in detail.

This architecture can be understood from the following figure:

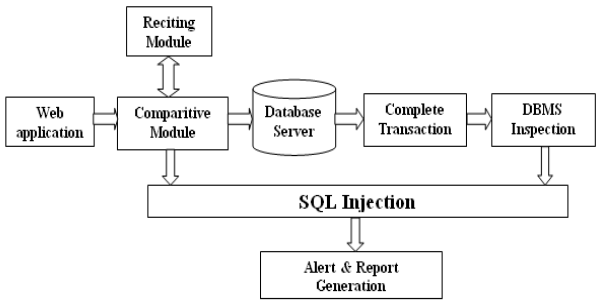

Figure 1: Architectural Design

A. Reciting Module

Reciting module includes the Meta strings library which comprises the predefined keywords and is updated with new type of information in terms of coordinates of the images, their pixels information, color resolution and the details on type of images (extension files like .jpg etc).

B. Comparative Module

In Comparative module, it gets an input from the web application and it compares the statement with the Meta strings library included in the Reciting Module, if founds any error message it attempts to block the query. It uses the Hirschberg algorithm to analyze the statement into set of characters. It uses Beschermen algorithm for comparison of each character with Meta Strings library and prevent SQL Injection attacks if found. The time complexity of this algorithm is O(nm) and space complexity is O(min(nm)).

8. Beschemen Algorithm

Beschermen algorithm is generally applicable algorithm for finding an optimal sequence alignment. Let, The Statement generated from the Web application is =Q The Character of the Statement is =C, The Meta String Library = M The Pixel Size of the statement= X The Original Statement included in the Meta String library= S.

Hence,
The Pixel size of the generated Statement = Q(X),
The Pixel size of the original String in Meta Strings Library = M(S(X)).

SQL INJECTION CODE

Select * from table where username=' r hul ' and pwd=123;
The algorithm describes the way how we follow the procedure for preventing the SQL Injection Attacks. Here, we had considered the generated query as Q, The Meta Strings Library is M, The Character of the Statement is C, The Pixel size of the statement as X and the Original statement to which the generated statement of web application is to be compared is represented as S. Let i & j be the position values of both the generated and actual statement. First, we consider the Statement Q from the Web application, and we choose Ci of Q. now the first character of the generated statement is compared with that of the actual statement included in the Meta Strings Library, in terms of their pixel information. If they are matched, i.e. Q(C(X)) = M(S(C(X))), then the character is considered and sent to the database server for the further transactions. This process repeats till the total statement is accepted.

If at any position, when Q(C(X))! = M(S(C(X))), then the character is considered as an image and the total statement should be blocked. Further the Details of the Prevention is to be reported. Comparing to Hirschberg algorithm, this approach is very advanced as it considers the images along with the strings for checking; where as Hirschberg principle is unable to check the images.

9. Result

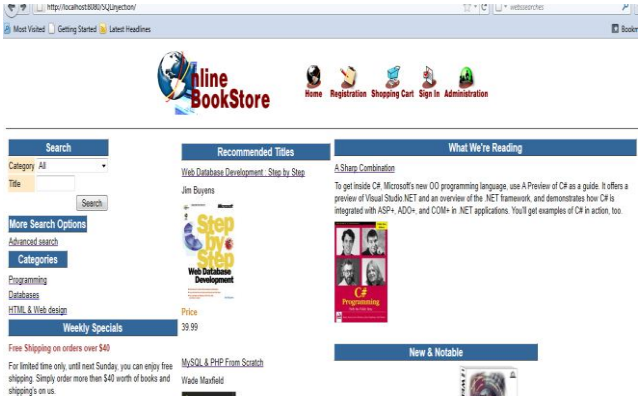

Figure 1: Home Page

Figure 2: Registration Page

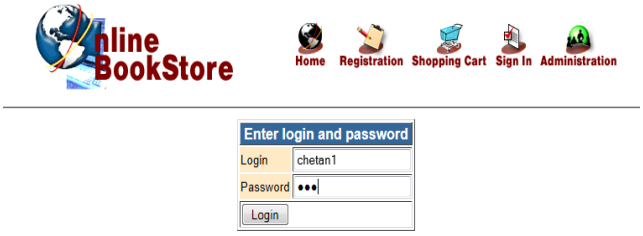

Figure 3: User Loging Page

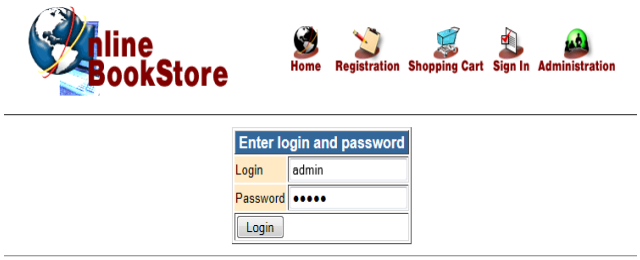

Figure 4: Admin Loging Page

Figure 5: Loging Page Used by Unauthorised User without SQL Code

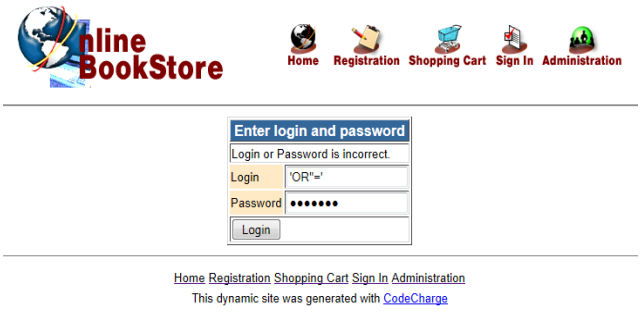

Figure 6: Loging Page Used by Unauthorised User with SQL
Code

Figure 7: Unauthorised User Acess Admin Page

10. Conclusion and Features

This project presented a highly automated approach for protecting Web applications from SQLIAs. Our approach consists of 1) Updating of Meta Strings library,2)using Beschermen algorithm to compare the given statement with updated meta strings library, find out the SQL Injection attacks. 3) Using DBMS Inspection methods to find out the transactions and reports the generation in case of SQL injection attacks. Beschermen algorithm is used to detect the SQL Injection attacks in order to reduce the time and space complexity and it provides the complete execution after analyzing the DBMS Inspection. Our approach also provides advantages over the many existing techniques environments, reduces the Time and Space complexities. Moreover, it requires no modification of the runtime system as it is defined at the application level, and hence imposes a low execution overhead.

Features

- For protecting Web applications (banking application).
- It also used in e-commerce application.
- Social networking sites to provide security on account related.

References

[1] R. Ezumalai, G.Aghila's "*Combinatorial approach for preventing SQL Injection attacks*", 2009 IEEE International Advance Computing Conference (IACC 2009 PanagiotisManolios, "WASP: Protecting Web Applications Using Positive Tainting and Syntax-Aware Evaluation", IEEE Transaction of Software Engineering Vol 34, Nol, January/February 2008.

[2] Konstantinos Kemalis and Theodora's Tzouramanis, "*Specification based approach on SQL Injection detection*", ACM, 2008.

[3] Stephen Thomas Patiala, India, 6-7, March 2009.

[4] William G.J. Hal fond, Alessandro Or so, and Laurie Williams "*Using Automated Fix Generation to Secure SQL Statements*", International workshop on Software Engineering and secure system ", IEEE, 2006.

[5] V. Benjamin Livshits and Monica S. Lam, "*Finding Security Vulnerabilities in Java Applications with Static Analysis*", ACM, 2005.

[6] Z. Su and G. Wassermann, "*The Essence of Command Injection Attacks in Web Applications*", 33rd ACM

SIGPLAN, SIGACT Symposium on Principles of Programming Languages, Charleston, South Carolina, USA, 2006, pp. 372-382.