Combined Approach to prevent XSS Attacks and SQL injection

Pankaj Sharma
Cert-In
Department of Electronics and IT
Govt of India
pankaj.vats@gmail.com

Rahul Johari
USICT, GGSIP University
DWARKA
DELHI
rahuljohari@hotmail.com

S.S Sarma
Cert-In
Department of Electronics and IT
Govt of India
ss.sarma@gmail.com

ABSTRACT
Multiple client side and server side vulnerabilities like SQL Injection and Cross Site Scripting (XSS) are discovered and exploited by malicious users. SQL injection Attacks (SQLIA) and Cross Site scripting vulnerabilities are top ranked in the Open Web Application Security Project (OWASP) top ten vulnerabilities list. Lots of security approaches are proposed like secure coding practices, encryption, static and dynamic analysis of code to secure the web applications but statistics shows that these vulnerabilities are still transpiring at the top. In this paper, we present an integrated model to prevent SQL Injection Attacks and Reflected Cross Site scripting attack in PHP based implementation. This Model is more effective to prevent SQL injection attack and Reflected Cross Site scripting attack in production web environment.

Keywords
XSS, AMNeSIA, SQLIA

1. INTRODUCTION
The Internet and web applications are playing very important role in our today's modern day life. Several activities of our daily life like browsing, online shopping and booking of travel tickets are becoming easier by the use of web applications. Most of the web applications use the database as a back end to store critical information such as user credentials, financial and payment information, company statistics etc. The websites that caters to these applications have been continuously targeted by highly motivated malicious users to acquire monetary gain. Our mechanism is divided into two modes, a safe mode and a production mode environment. In the safe mode we construct a security query model for SQL injection and sanitizer model for Reflected Cross Site scripting attack for each identified SQL queries and input entry points for SQL injection and Reflected Cross Site scripting respectively. In the production environment, input entries which create dynamic SQL queries are validated against security query model generated in safe mode and normal input text entered by the user is validated by sanitizer model instrumented in the code at safe mode. The results and analysis shows that the proposed approach is simple and effective to prevent common SQL injection vulnerabilities and Reflected Cross Site Scripting vulnerabilities.

2. RELATED WORK

MHAPSIA (Static cum Runtime Monitoring Approach): This approach focuses on static analysis and run time validation i.e. it runs the application in two different modes safe and real. In the safe environment, it creates a query model for all legitimate SQL statements using a DFA (Deterministic Finite Automata). Nodes of the NDFA are SQL keywords and operators with special symbols. In the real environment, SQL statements are intercepted with the instrumented code and then validated with the query model generated in the safe environment. A limitation of this approach is that the particular SQL injection or 1=1 # is not blocked by the query model as the query model for this injection is same as that of a legitimate query model.

AMNeSIA (JSP approach): Similarly, Model Based Hybrid Approach (AMNeSIA) prevents SQL injection attacks by forming a query model which is again an NDFA based on the same construction pattern as that of the above approach. An inadequacy of Amnesia tool is that, it can only be used in JSP web applications. This tool makes use of the JSA library to construct a query model which is not available in any other language.

3. LIMITATIONS OF EXISTING APPROACHES
- The major drawback of MHAPSIA is that it doesn’t prevent all the SQL injection attacks. MHAPSIA is not designed for preventing Reflected Cross Site Scripting Vulnerabilities.
- Query models are stored in run time arrays only there is no permanent source of storage due to which whenever the application is run, safe mode query models need to be constructed each time.

4. PROPOSED SOLUTION
The proposed approach is implemented in various PHP applications. In this section, we briefly the working of the proposed tool works in two different modes, safe and production mode. Initially, it starts with verifying the application for SQL and XSS vulnerability for which it takes the complete application path and scans all files of web application for SQL queries and input entry points with respect to an array vulsig () array. Then the tool enters into the instrumentation phase of the safe mode which is further divided into two modules, ispot identification and instrumentation. In ispot identification, each location verified in the verification module is assigned an ispot id and ispot line number. Further the instrumentation module is divided into two modules, which are:
1. Instrumentation for SQL injection.
The output of this module is an instrumented file which contains a validator program instrumented before each SQL query located by the ispot. Then these instrumented files are executed in the safe mode in which Security Query model is generated for all legitimate inputs.

2. Instrumentation for XSS:
The output of this module is an instrumented file which contains a sanitizer program instrumented after each input entry points. This function sanitizes all malicious scripts entered by the any attacker or malicious user. In the safe mode, the executions of the instrumented files are not mandatory.

5. RESULT AND ANALYSIS:
The two tables show the various open web applications that are tested against this integrated tool to prevent SQL Injection and Reflected Cross Site Scripting Attacks. Table 1 shows the detailed analysis for XSS attacks and table 2 shows the detailed analysis for SQL injection attack. We have analysed from the results that the proposed integrated model covers almost all kinds of SQL injection and XSS attacks. The effectiveness of the proposed approach is determined by the ratio of the number of attacks prevented to the total number of attacks performed. Results show that the proposed approach is 100% effective to prevent SQL injection and Reflected Cross Side Scripting attacks.

Table 1. Detailed Execution Analysis Results for Proposed Model for XSS Attack

<table>
<thead>
<tr>
<th>Application</th>
<th>Lines Of Code(K)</th>
<th>Ispot Instrumented for XSS</th>
<th>Detection Rate (%)</th>
<th>Instrumentation Overhead (%)</th>
<th>False Positive</th>
<th>Query Execution Overhead (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Online Bookstore</td>
<td>4.3</td>
<td>26</td>
<td>100</td>
<td>7</td>
<td>13</td>
<td>1.23</td>
</tr>
<tr>
<td>Matrimonial</td>
<td>3.7</td>
<td>23</td>
<td>100</td>
<td>5</td>
<td>11</td>
<td>1.10</td>
</tr>
<tr>
<td>Student Portal</td>
<td>8.2</td>
<td>30</td>
<td>100</td>
<td>6</td>
<td>18</td>
<td>2.03</td>
</tr>
<tr>
<td>Travel portal</td>
<td>9.9</td>
<td>40</td>
<td>100</td>
<td>10</td>
<td>20</td>
<td>2.04</td>
</tr>
</tbody>
</table>

Table 2. Detailed Execution Analysis Results For Proposed Model For SQL Attack

<table>
<thead>
<tr>
<th>Application</th>
<th>Lines Of Code(K)</th>
<th>Ispot Instrumented for SQL Injection</th>
<th>Detection Rate (%)</th>
<th>Instrumentation Overhead (%)</th>
<th>False Positive</th>
<th>Query Execution Overhead (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Online Bookstore</td>
<td>4.3</td>
<td>26</td>
<td>100</td>
<td>7</td>
<td>13</td>
<td>1.23</td>
</tr>
<tr>
<td>Matrimonial</td>
<td>3.7</td>
<td>23</td>
<td>100</td>
<td>5</td>
<td>11</td>
<td>1.10</td>
</tr>
<tr>
<td>Student Portal</td>
<td>8.2</td>
<td>30</td>
<td>100</td>
<td>6</td>
<td>18</td>
<td>2.03</td>
</tr>
<tr>
<td>Travel portal</td>
<td>9.9</td>
<td>40</td>
<td>100</td>
<td>10</td>
<td>20</td>
<td>2.04</td>
</tr>
</tbody>
</table>

6. ACKNOWLEDGMENTS

The author express sincere gratitude and thanks to administration of Cert-In and GGS IP University for providing academic and research oriented environment.

7. REFERENCES


