2012 International Conference on Communication Systems and Network Technologies

A Survey On Web Application Vulnerabilities(SQLIA,XSS)Exploitation and Security Engine for SQL Injection

Rahul Johari
USIT, GGSIP University
Sector 16-C Dwarka
Delhi, India
rahuljohari@hotmail.com

Pankaj Sharma
CERT-In,
Ministry of Communications and IT
Govt. of India
Delhi India
Pankaj.vats@gmail.com

Abstract— **Today almost all organizations have improved their performance through allowing more information exchange within their organization as well as between their distributers, suppliers, and customers using web support. Databases are central to the modern websites as they provide necessary data as well as stores critical information such as user credentials, financial and payment information, company statistics etc. These websites have been continuously targeted by highly motivated malicious users to acquire monetary gain. Structured Query Language (SQL) injection and Cross Site Scripting Attack (XSS) is perhaps one of the most common application layer attack technique used by attacker to deface the website, manipulate or delete the content through inputting unwanted command strings. Structured Query Language Injection Attacks (SQLIA) is ranked 1st in the Open Web Application Security Project (OWASP) [1] top 10 vulnerability list and has resulted in massive attacks on a number of websites in the past few years. In this paper, we present a detailed review on various types of Structured Query Language Injection attacks, Cross Site Scripting Attack, vulnerabilities, and prevention techniques. Besides presenting our findings from the survey, we also propose future expectations and possible development of countermeasures against Structured Query Language Injection attacks.**

Keywords- SQL Injection Attack, Static Analysis, Dynamic Analysis, Web Vulnerabilities, Unauthorized Access, Authentication Bypass, Input Validation, Database Mapping etc.

I. INTRODUCTION

SQL Injection Attacks (SQLIAs)-Structured Query Language (SQL) is an interpreted language used in database driven web applications which construct SQL statements that incorporate user-supplied data or text. If this is done in an unsafe manner, then the web application may be vulnerable to SQL Injection Attack i.e. If user supplied data is not properly validated then user can modify or craft a malicious SQL statements and can execute arbitrary code on the target machine or modify the contents of database.

Problem formalization- Any web application can be formalized with respect to SQL Injection Attack as follows:

- It accepts the input from user or system.
- It concatenates input with hardcoded SQL statement and builds complete query structure.
- Query generated gets executed and concatenates result with HTML code.

In the context of above formalization SQL Injection Attack is targeted on a program at the database layer which is connected to a web application. This SQL Injection Attack exploits weakness or vulnerability in the target program to properly verify the input supplied to it through a web form. The Common Weakness Enumeration (CWE) framework [2] which provides unified set of software weaknesses defined [3]SQL injection weakness as “not neutralizing or incorrectly neutralizing special elements that could modify the intended SQL command”. In a typical SQL injection attack the attacker posts specially crafted Structured Query Language (SQL) statements which are executed in the database server and produce malicious outcomes.

II. SQL INJECTION BACKGROUND

A. *Why SQL Injection is a major Threat?*

The vulnerability trends indicate that significant portion of the vulnerabilities are in applications. Further Cross site scripting and SQL injection are prominent application vulnerabilities among those reported in applications. The SQL injection attacks pose greater risk due to the fact that they impact databases which are critical to any organization. From response perspective also, the remedial action need to be taken by the developer or programmer since the flaw need to be corrected by code level changes. This requires comparatively longer times to take corrective actions after SQL injection vulnerabilities are detected. The attack trends also indicate that SQL injection vulnerabilities are exploited during recent years, in mass scale on vulnerable web applications. Mass scale web intrusions were carried out by “ASPROX” botnet during 2008 and 2010 which resulted in infection of number of websites in a short span of time. ASPROX used specially formed malicious SQL queries to infiltrate vulnerable databases. In a typical attack ASPROX used Google queries to harvest vulnerable ASP pages and carried out SQL injection and subsequently inserting iFrame links into databases. These iFrames were used to conduct drive-by-download attacks wherein visitors of affected websites are redirected to malicious websites which are used to propagate malware on to users' systems. Hence, SQL Injection could be very dangerous in many cases depending on the platform where the attack is launched and it gets success in injecting rogue users to the target system.

B. *Impact of SQL Injection*

As we already mentioned SQL injection attack is accomplished by providing data (inclusion of SQL queries) from an external source which is further used to dynamically construct a SQL query. The impact and consequences of SQL injection attacks can be classified as follows:

1) Confidentiality: Loss of confidentiality is a major problem with SQL Injection attacks since SQL databases generally hold sensitive and critical information which could be viewed by unauthorized users as a consequence of successful SQL injection attack.

2) Integrity: Successful SQL injection attack allows external source to make unauthorized modifications such as altering or even deleting information from target databases.

3) Authentication: Poorly written SQL queries do not properly validate user names and passwords, which allows unauthenticated entity or attacker to connect to the affected database or application as an authenticated user, without initial knowledge of the password or even user name.

4) Authorization: Successful exploitation of SQL injection vulnerability, allows attacker to change authorization information and gain elevated privileges if the authorization information is stored in the affected database

In real world scenario, it is very hard to detect the SQL injection prior to its impact. In most number of scenarios, unauthorized activity is performed by the attacker through valid user credentials or by using inherent features of database application such as malicious modification of existing SQL Queries of web application that are accessing critical sections of the affected databases.

III. CROSS SITE SCRIPTING (XSS)

Cross-site Scripting (XSS) is another common web application attack technique that involves echoing attacker-supplied code into a user's browser instance or client via web pages viewed by the target users. Here the attackers host or inject attack code written in different static or dynamic contents such as HTML, Java, JavaScript, ActiveX, Flash or any other browser supported technology.

When an attacker gets a user's browser to execute his/her script, the script will run within the security context (or zone) of the hosting web site. With this kind of privilege, the application code has the ability to read, change and transmit any critical data accessible by the browser. Successful XSS attack allows attackers to hijack user's account via cookie, redirecting user to another website from the website visited and there by facilitating other types of attacks such as phishing or drive-by-download attacks. XSS attack poses significant risk in cases where the browser interacts closely with file system on the user's computers for loading content. XSS attacks are commonly used to hijack sessions of users visiting websites facilitating e-Commerce, wherein malicious script/code runs on user's client and captures cookie information of user's browser allowing hijacking of session.

There are different types of XSS attacks, primarily two types called "persistent" and "non-persistent" XSS are mentioned here. The difference between the two types lie in the way the server side and client side vulnerabilities are exploited.

A. *Stored or Persistent attacks*-

Occurs when the malicious code is submitted to a web application where it's stored permanently. Examples of an attacker's favorite targets often include web mail messages, and forum posts. When a user visits the polluted webmail message or forum , the script/code executes in the user's context. This attack scenario does not require enticing targeted user to click on any URL.

B. *Reflected or Non-persistent attacks*

Occurs when the server does not properly sanitize the output server to a visiting web browser/client. In typical attack scenario, the attacker target visitor of a specific website 'abc.com' containing reflected XSS vulnerability and tricks targeted user to click on a maliciously crafted URL. The user intend to visit the 'abc.com' and does so but in the process the client side script/code contained in the malicious URL supplied by the attacker executes thereby enabling the attacker to gain access to sensitive user credentials, cookies etc.

C. *Impact of Cross-site Scripting Attack(XSS)*-

The impact of Cross-site scripting attack varies depending up on the level of access and type of information gained by the attacker. The consequences of successful XSS attack range from annoyance to compromise of user's credentials. Typically XSS attacks are used as part of larger scheme of attacks such as redirection of visitors of a trusted website to malicious websites, malware propagation, e-Commerce frauds etc.

IV. SQL INJECTION DETECTION AND PREVENTION

In order to prevent SQL Injection attacks many existing techniques, such as Content filtering, penetration testing, and defensive coding, can be used to detect and prevent a subset of the SQL Injection Vulnerabilities. Over the past years, there has been plenty of research going on in the both academic institutes as well as industries to prevent injection attacks. Following are some prevention mechanism proposed by researchers which has been seen as more effective one.

A. "An Authentication Scheme using Hybrid Encryption " (Indrani Balasundaram, E.Ramaraj)-(2011)

[4]Indrani Balasundram and E.Ramaraj proposed an authentication scheme in which they propose an algorithm which uses both Advance Encryption Standard (AES) and Rivest-Shamir-Adleman(RSA) to prevent SQL injection attacks. In this method a unique secret key is fixed or assigned for every client or user. On the server side server uses private key and public key combination for RSA encryption. In this method, two level of encryption is applied on login query:

- To encrypt user name and password symmetric key encryption is used with the help of user's secret key.
- To encrypt the query the scheme uses asymmetric key encryption by using public key of the server.

The whole procedure completed in three phases :

- Registration Phase
- Login Phase
- Verification Phase

The proposed scheme is very efficient, it needs 961.88ms for encryption or decryption and this can be negligible.

Some disadvantages also exist with this approach:

- It is not made for URL based SQL injection attacks.
- It is Very difficult to maintain every user secret key at server side and client side.
- There is no security mechanism at registration phase.

B. "Effective SQL Injection Attack Reconstruction Using Network Recording"(Allen Pomeroy and Qing Tan)(2011)

[5]Allen Pomeroy and Qing Tan has suggested a technique for finding vulnerabilities in Web Application such as SQL injection attack by network recording. In this approach network forensic techniques and tools are used to analyze the network packets containing get and post requests of a web application. This approach uses network based Intrusion Detection System (IDS) to trigger network recording of suspected application attacks.
Some disadvantages also exist with this approach:

- Difficult to record high volume traffic.
- Packet fragmentation attack could bypass this approach.

C. "Insecure Query Processing in the Delay/Fault Tolerant Mobile Sensor Network (DFT-MSN) and Mobile Peer to Peer Network"(Rahul Johari and Neelima Gupta)(2011)

[6]Rahul Johari and Neelima Gupta proposed set of SQL/TIQL queries that are exchanged between the pair of nodes in the DFT-MSN and MP2PN.They also portrays their execution on Oracle 9i Enterprise Edition Release 9.2.0.1.0 Production and expose how these SQL queries are vulnerable to the SQL Injection Attack which can either be launched manually or through the various proprietary and open source SQL Injection tools.

D. "Secure Query Processing In Delay Tolerant Network Using Java Cryptography Architecture"(Rahul Johari and Neelima Gupta)(2011)

[7]Rahul Johari and Neelima Gupta proposed a lightweight cryptography authentication mechanism at the source node and destination nodes of Delay Tolerant Network to prevent unauthorized modification of SQL queries during bundle transfer between nodes. The proposed method uses message digest (MD5) Hashing algorithm for encryption of data stream before it is transmitted via multiple intermediate nodes so as to reach to the destination node.

E. " SQL Injection Attack Detection using the Removal of SQL Query Attribute Values"(Jeom-Goo Kim)(2011)

[8]Jeom-Goo Kim presents a effective approach of removal of SQL query passed by user in SQL query attributes values. This approach uses combined static and Dynamic Analysis. The proposed method will utilize a function which has the capability to detect the attribute values of static SQL query in web application. This function also detected the SQL queries generated at runtime. This approach profile the SQL query generated from normal users and compare this with SQL query generated dynamically by the attacker.

Some disadvantages also exist with this approach are:

- Developer learning is required.
- Source code adjustment is needed .

F. "Dynamic Candidate Evaluations Approach to prevent SQL injection"(P. Bisht, P. Madhusudan, and V. N. Venkatakrishnan)(2010)

[9]Prithvi Bisht and his team members propose a tool called Candidate evaluation for Discovering Intent Dynamically (CANDID).This method record the programmer-intended SQL query structure on any input(candidate inputs) from the legitimate user and compare this with the query structure generated with the attackers input.

Some disadvantages also exist with this approach are:

- Developer learning is required.
- It is not possible to make a complete set of legitimate inputs for a large web application.

G. "Obfuscation-based Analysis of SQL Injection Attacks"(Raju Halder and Agostino Cortesi)(2010)

[10]In this method an obfuscation/deobfuscation based technique is proposed to detect SQL Injection Attacks (SQLIA) in a SQL query before sending it to database. this technique has three phases:

- Static phase: In the static phase, the SQL Queries in the web application code are replaced by queries in obfuscated form.

- Dynamic Phase: In this phase user inputs are merged with the obfuscated query at run-time. After merging dynamic verifier checks the obfuscated query at atomic formula level to detect the SQL injection attack.

- If no SQL injection found during the verification phase reconstruction of the original query from the obfuscated query is carried out before submitting it to the database.

H. "SQL injection Detection via Automatic Test Case Generation of Programs" (Michelle Ruse, Tanmoy Sarkar, Samik Basu)(2010).

[11]This approach uses automatic test case generation to detect SQL Injection Vulnerabilities. The main idea behind this framework is based on creating a specific model that deals with SQL queries automatically. It also captures the dependencies between various components of the query. The used CREST(Automatic Test Generation Tool for C) test-generator and identify the conditions in which the queries are vulnerable. Based on the results, the methodology is shown to be able to specifically identify the causal set and obtain 85% and 69% reduction respectively while experimenting on few sample examples.

I. "Combinatorial Method for Preventing SQL Injection Attacks" (R. Ezumalai, G. Aghila) 2009

[12]This approach uses both static and dynamic approach to detect SQL injection. It is a signature based SQL injection detection technique. In this approach they generate hotspots for SQL queries in web application code and divide these hotspots into tokens and send it for validation where it uses Hirschberg's algorithm, which is a divide and conquer version of the Needleman-Wunsch algorithm, used to detect SQL injection attacks. Since, it is defined at the application level, requires no change in the runtime system, and imposes a low execution overhead.

J. "An Approach for SQL Injection Vulnerability Detection- AMNeSIA"(M. Junjin)(2009)

[13]Analysis and Monitoring for NEutralizing SQL-Injection Attacks (AMNeSIA) is a fully automated technique for detecting and preventing SQL injection attacks. It works in two phases.

- Static analysis: In this phase it analyze web application code and automatically generate the SQL query mode on the basis of possible legitimate queries.

- Runtime analysis: In this phase it scan all dynamically generated SQL queries and checks them to be with compliance to the statically generated models in the previous step. When this step detects that a query is not matched with the query model, it classifies the input as an SQL injection attack, logs the necessary information and throws an predefined exception that the application can then deal with suitably.

Principle behind AMNEeSIA*:*

- *Generation Hotspots:* In this step the tool analyzing the web application code by simply scanning the code and identifying the hotspots which are the locations where the SQL queries are sent for execution to the databse.
- *SQL query model:* The Application class files are given as input to JSA (Java String Analyzer). It outputs a NDFA (Non-Deterministic Finite Automata) with all character level possibilities of the strings. It further analyzes the NDFA to create a SQL query model and change it into semantically meaningful SQL keywords, operators and literal values.
- *Instrumention of the Web Application:* In this phase, AMNeSIA creates a function that checks for queries at runtime. It inserts the call to the a function before accessing the database at each place. The function call is invoked with two parameters; the SQL query that is to be submitted and the Hotspot id.
- Runtime Validation: During runtime, the query is sent to the runtime validation before accessing the database. The validator program parses the SQL query string into a sequence of tokens. Validator program verifies for query acceptance test by traversing the sequence of tokens. Depending on test result, it allows/restrict the database access.

Limitations and Assumptions of AMNeSIA model in JSP:

There is a chance to get false positives and false negatives in some situations.

- When the string analysis results in a SQL query model that is overly conservative and includes spurious queries (i.e. queries that could not be generated by the application) that happen to match an attack.

- When a legitimate query happens to have the same" SQL structure" of an attack. For example, if a developer adds conditions to a query from within a loop, an attacker who inserts an additional condition of the same type would generate a query that does not violate the SQL-query model

- In some cases, as the analysis cannot distinguish a variable or a hard-coded SQL token, it raises false positives for a string model that is precise enough. In particular, if the hard-coded string is used in the application to construct a SQL token, the technique will generate an incomplete SQL-query model.

AMNeSIA tool makes use of Java String Analyzer (JSA) library to statically analyze the source code and thereby get the query models. Thus it can't be used for web applications other than those built on JSP such as PHP or ASP

K. "Automated generation of prepared statement to remove SQL injection vulnerabilities"(Stephen Thomas , Laurie Williams, Tao Xie) (2008)

[14]Stephen Thomas presents an algorithm in which prepared statement in SQL queries are replaced by secure prepare statements for removing SQL vulnerabilities .Prepared statements have a static structure, which prevents SQL injection attacks from changing the logical structure of a prepared statement. In this approach they created a algorithm which replaces prepared statement and a corresponding tool for automated fix generation. Then conducted four case studies of open source projects to evaluate the capability of the algorithm and its automation. The results show that algorithm correctly replaced 94% of the SQL injection vulnerabilities in these projects.

V. CROSS SITE SCRIPTING (XSS) DETECTION AND PREVENTION

A. "Protecting Cookies from Cross Site Script AttacksUsing Dynamic Cookies Rewriting Technique". (Rattipong Putthacharoen, Pratheep Bunyatnoparat) (2011)

[15] This approach aims to change the cookies in such a way that they will become useless for XSS attacks. This technique is called "Dynamic Cookie Rewriting" implemented in a web proxy where it will automatically replace the cookies with the randomized value before sending the cookie to the browser. In this way browser will keep the randomized value instead of original value sent by the web server. At the web server end the return cookie from the browser again rewritten to its original form at the web proxy before being forwarded to the web server. So in case if XSS attacks steal the cookies from the browser's database, the cookies cannot be used by the attacker to impersonate the users. This technique is not tested with HTTPs connections.

B. "An Execution-flow Based Method for Detecting Cross-Site Scripting Attacks"(Qianjie Zhang, Hao Chen, Jianhua Sun)(2010)

[16]Qianjie Zhang, Hao Chen, Jianhua Sun presents an execution-flow analysis for JavaScript programs running in a web browser to prevent Cross-site Scripting (XSS) attacks.In this approach they use Finite-State Automata (FSA) to model the client-side behavior of Asynchronous JavaScript and XML (AJAX) applications under normal execution. In this method system is deployed in proxy mode. In this mode the proxy analyzes the execution flow of client-side JavaScript and checks them to be with compliance to the models generated by FSA. It stops potentially malicious scripts, which do not conform to the FSA before the requested web pages arrive at the browser. This method is evaluated against many real-world web applications and the result shows that it protects against a variety of malicious scripts to prevent XSS attacks and has an acceptable performance overhead.

C. "Automatic Creation of SQL Injection and Cross-site Scripting (XSS) Attacks(Ardilla)" (Adam Kie˙zun, Philip J. Guo, Karthick Jayaraman, Michael D. Ernst)(2010)

[17] Adam Kie˙zun has suggested a technique for finding vulnerabilities in Web Application such as SQL injection attack and Cross site scripting (XSS).They implement this technique as an automated tool called Ardilla. This method uses static code analysis to find vulnerabilities. This technique works source code of the application , creates concrete inputs that expose vulnerabilities and operates before the application is deployed. It analyses application internals to discover vulnerabilities in the code. It is based on input generation, taint propagation, and input mutation to find variants of an execution that exploit vulnerability. This tool is designed for php applications.

Some disadvantages also exist with this approach are:

- Developer availability and learning is required.
- Source code adjustment is needed.
- If the original developer left the project it is very difficult to patch the vulnerabilities.
- It is tested on PHP based applications.

VI. CONCLUSIONS

From the survey of various articles/papers it is found that SQL Injection and Cross-site Scripting (XSS) Attacks are most powerful and easiest attack methods on the Web

Application. This study presents a survey of current techniques for defending against SQL injection and XSS exploits. Our review finds that existing techniques suffer from one or more of the following five weaknesses:

- Inherent limitations
- Incomplete implementations
- Complex frameworks
- Runtime overheads
- Intensive manual work requirements
- False positives and false negatives

The academic field and industry has developed promising strategies such as AMNeSIA , ARDILLA etc. products which are rising and likely to become essential parts of comprehensive online data protection strategies. Yet, despite the effectiveness of these products, we believe that their use should not excuse developers from applying preventive coding techniques, as these hold true potential when implemented properly. Keeping in view the emerging web technologies and extensive usage of highly interactive content over Internet, it is imperative for the software development houses and developers to frame and follow appropriate security framework to build security during Software Development Life Cycle.

ACKNOWLEDGEMENT

I wish to express sincere gratitude to the administration of Department of Information Technology Govt. of India and GGSIP University for providing the academic environment to pursue research activities. In particular I would like to thank Mr. S. S. Sarma, Scientist 'E' (CERT-In) for inputs.

REFERENCES

[1] http://www.owasp.org/index.php/Top_10_2010-A1-Injection, retrieve on 13/01/2010

[2] http://cwe.mitre.org/index.html

[3] http://cwe.mitre.org/data/definitions/89.html

[4]- Indrani Balasundaram, E.Ramaraj "An Authentication Scheme for Preventing SQL Injection Attack Using Hybrid Encryption(PSQLIA-HBE"(ISSN 1450-216X Vol.53 No.3 (2011),pp.359-368)

[5]Pomeroy, A Qing Tan Sch. of Comput. & Inf. Syst., Athabasca Univ., Athabasca, AB, Canada " Effective SQL Injection Attack Reconstruction Using Network Recording" in Computer and Information Technology (CIT), 2011 IEEE 11th International onference Issue Date: Aug. 31 2011-Sept. 2 2011 On page(s): 552 - 556

[6 Johari R.,Gupta N., "Insecure Query Processing in Delay/Fault Tolerant Mobile Sensor Network(DFT-MSN) and Mobile Peer to Peer Network" accepted by Springer (LNCS) in Communications in Computer and Information Science (CCIS) Series,Chennai (July 2011).

[7] Johari R.,Gupta N., "Secure Query Processing in Delay Tolerant Network Using Java Cryptography Architecture". 2011 IEEE Computational Intelligence and Communication Networks (CICN) Gwalior, India, 7-9 Oct. 201

[8]Jeom-Goo Kim; Dept. of Comput. Sci., Namseoul Univ., Cheonan, South Korea " Injection Attack Detection using the Removal of SQL Query Attribute Values" in Information Science and Applications (ICISA), 2011 International Conference Issue Date: 26-29 April 2011, On page(s): 1 - 7

[9] P. Bisht, P. Madhusudan, and V. N. Venkatakrishnan. "CANDID:Dynamic Candidate Evaluations for Automatic Prevention of SQL Injection Attacks". ACM Trans. Inf. Syst. Secur., 13(2):1–39, 2010..

[10] Raju Halder and Agostino Cortesi, "Obfuscation-based Analysis of SQL Injection Attacks". 978-1-4244-7755-5/10/$26.00 ©2010 IEEE

[11] M. Ruse, T. Sarkar and S. Basu. "Analysis & Detection of SQL Injection Vulnerabilities via Automatic Test Case Generation of Programs." 10th Annual International Symposium on Applications and the Internet pp. 31 – 37 (2010)

[12] R. Ezumalai, G. Aghila, Combinatorial Approach for Preventing SQL Injection Attacks. 2009 IEEE International Advance Computing Conference (IACC 2009) Patiala, India, 6-7 March 2009

[13] M. Junjin, "An Approach for SQL Injection Vulnerability Detection," Proc. of the 6th Int. Conf. on Information Technology:New Generations, Las Vegas, Nevada, pp. 1411-1414, April 2009.

[14]Stephen Thomas *, Laurie Williams, Tao Xie" On automated prepared statement generation to remove SQL injection vulnerabilities "Department of Computer Science, Box 8206, North Carolina State University, Raleigh, NC 27695, USA

[15] Rattipong Putthacharoen, Pratheep Bunyatnoparat " Protecting Cookies from Cross Site Script Attacks Using Dynamic Cookies Rewriting Technique" Feb. 13~16, 2011 ICACT2011.
Method for Detecting Cross-Site Scripting Attacks".

[16] Qianjie Zhang, Hao Chen, Jianhua Sun "An Execution-flow Based Method for Detecting Cross-Site Scripting Attacks" China(2010)

[17] " Automatic Creation of SQL Injection and Cross-Site Scripting Attacks "ARDILLA(Adam Kie˙zun, Philip J. Guo, Karthick Jayaraman, Michael D. Ernst)