International Journal of Multidisciplinary Research and Development

Volume: 2, Issue: 5, 375-379
May 2015
www.allsubjectjournal.com
e-ISSN: 2349-4182
p-ISSN: 2349-5979
Impact Factor: 3.762

Naveesha Saharan
Student, M.Tech. FET, Mody University of Science and Technology Lakshmangarh, Sikar, Rajasthan, India

Aditi Kajala
Assistant Prof. FET, Mody University of Science and Technology Lakshmangarh, Sikar, Rajasthan, India

Correspondence:
Naveesha Saharan
Student, M.Tech. FET, Mody University of Science and Technology Lakshmangarh, Sikar, Rajasthan, India

SQL Injection and Proposed Methods: Comparison of tools and efficiency for preventing SQL attacks

Naveesha Saharan, Aditi Kajala

Abstract
SQL Injection continues to be the topmost security risk in the world according to Open Web Application Security Project (OWASP) [1] top 10 vulnerability list of 2013. The ease of exploitability and severe impact are what that lands this attack at top spot. As the countermeasures are made more sophisticated, injection also continues to evolve itself, thus thwarting the attempt to completely tackle this attack. The vulnerable data is a source of worry for government as well as financial institution. In this digital age, the consequence of an unauthorized access to a database could vary from lack of confidentiality of clients to a destabilized nation embroiled in war. In this paper, a detailed survey of types of SQL Injection and proposed methods and theories is been presented along with tools and their efficiency in intercepting and thus preventing SQL attacks.

Keywords: SQL injection, Union query, Piggy-backed query

Introduction
In today's age, all work is done online provided the flexibility and portability of web applications. The ease of accessibility through these web applications has completely revolutionized the traditional view of an office or a company. The data are stored in databases which can be accessed anywhere and anytime through a network. These databases are built on basis of Codd's principle which uses SQL (pronounced as "sequel") to interact with external environment. The standard format which is followed for all databases has improved consistency but at the cost of ease of exploitability. The staggering amount of data present on net has also led to compounded security threat. This vulnerable data in case of malicious access could cause incalculable financial loss along with irreparable damage to one's reputation.

Overview of SQL injection
Structured Query Language (SQL) is a high level language used in database management systems (DBMSs). SQL was originally developed in the early 1970's by Edgar F. Codd at IBM. It allows the user to modify, delete or just access data. The "query" is unit of execution in SQL which returns a set of rows and columns when satisfies the condition specified in query.
In database driven web applications, SQL statements incorporate user-supplied data or text. If insertion of user supplied data is done is done in an unsafe manner, then the web application becomes vulnerable to SQL Injection Attack. SQL-Injection vulnerabilities and attacks occur between the Presentation tier and the CGI tier. Most of vulnerabilities are accidentally made in the development stage. The data flow of each tier using malicious input data are as shown in below figure. It depicts the user's authentication step. When an authenticated user enters its ID and Password, the Presentation tier uses the GET and POST method to send the data to the CGI tier. The SQL query within the CGI tier connects to the database and processes the data.

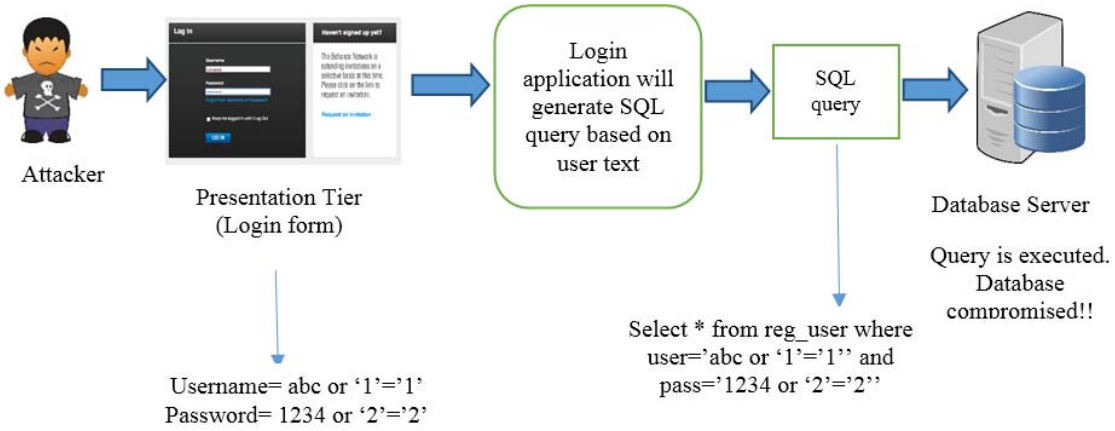

Fig: 1 Concept of SQL Injection

SQL Injection Attacks

In this section, we demonstrate the type of vulnerabilities in programming which leads to various types of SQL Injection attacks depending on type of vulnerability exploited.

Table: 1 Types of Vulnerabilities

Type of vulnerability	Description
Type 1	Unclear distinction between data types accepted as input
Type 2	Delay of operation analysis till the runtime phase thus consideration of current variables instead of source code operation
Type 3	Improper type specification while designing
Type 4	Input validation is not well defined and incorrect analysis of sanitized inputs

Table: 2 Summary of types of attack

Type of attack	Attacker's aim	Description	Example
Tautologies	Bypassing authentication and extracting data	Conditional statements are formed in such a way that they are always true.	Select * from emp_info where empid=" or '7=7';
Logically Incorrect queries	To extract information about database and identify injectable patterns and	Invalid queries are executed leading to error messages which constitute information about data type or table name.	Aggregate functions applied on varchar or invalid datatypes Or using 'having' and 'group by' clauses.
Union Query	Bypassing authentication and extracting data	By using operator 'union', malicious query is joined with safe query	Select * from user where user='ravi' union select * from admin where id='3142'--'pass='2=2';
Stored procedure	Privilege escalation, executing remote commands, DoS	Using built-in procedures, malicious actions are performed.	Commands like DROPTABLE, SHUTDOWN are executed.
Piggy-backed queries	Data extraction and modification, DoS	Malicious query is appended to legitimate query. On execution of first query, second also gets executed	Select * from user where name= 'ravi' and pass='1234';drop table user;
Alternate Encodings	To evade detection	Some database have filters which detect characters like --, % etc as bad character. So to avoid detection, attacker encodes the query in ASCII or Unicode.	SELECT salary FROM users WHERE login=" AND pin=0; exec (char (0x73687574646j776e));
Inference Blind injection Timing attacks	Data extraction, database schema discovery and identification of injectable patterns	Logical conclusions are drawn on basis of true/false questions. Database schema is guessed by gathering responses on basis of true/false questions. Information collection is done through observing response time taken in answering questions	Attackers injects query to discover the vulnerabilities like select * from user where id='12' and pass='1=0'; to check if there is input validation or not. Keywords like waitfor are inserted to delay execution if query is true etc.

SQL Injection Detection and Prevention

In order to prevent SQL Injection attacks many techniques have been proposed. These techniques include defensive coding, encryption and obfuscation, static binding, dynamic binding etc. The methods and theories proposed varies from introducing middleware to developing completely different algorithm for detection and prevention. Concepts from

cryptography, Machine Learning, query translation etc. are developed.

Below are some methods which have gained attention in past years.

A. "Integrated approach to prevent SQL Injection attack and reflected cross site scripting attack"(Pankaj Sharma, Rahul Johari and S.S Sharma)(2012)

[2]Pankaj Sharma, Rahul Johari and S.S Sharma proposed query model generator using **hybrid approach**, in which they proposed algorithms by extending existing **MHAPSIA** model (model based hybrid approach to prevent SQL injection attacks in PHP-Kunal et al. 2011) for PHP applications by incorporating the logic to thwart SQL injection attacks and integrated a scheme to avert reflected cross site scripting attacks.

The proposed model works in two modes, safe mode and production mode; and consists of following modules:

- Verification module, scans all files of web application to locate all SQL query signatures for SQL injection attack and input type html entities for reflected cross site scripting attacks.
- Ispot identification module identifies all the ispots which are basically the verified locations of signatures found in above module.
- Instrumentation module is further divided into two sub modules i.e. Instrumentation module for SQL Injection and Instrumentation module for XSS. It generates instrumented files after appending instrumented function.
- Query Model Construction module, runs the whole instrumented web application is safe mode by taking legitimate inputs. The model generated for legitimate inputs is stored and then used for validation in real mode.
- Validation module, called in production mode, wherein actual inputs are validated against the static query model generated in safe mode.

Disadvantages

- The analysis is performed for known attacks only, thus model needs to be tested for zero day exploits.
- Strength of solution against complex and exhaustive applications is not yet confirmed.

B. "SQLIMW: a new mechanism against SQL-Injection "(Gao Jiao, Chang-Ming XU, Jing Msohua)-(2012)

[3]Gao Jiao and et al proposed a new mechanism to prevent SQL injection by adding a middleware, SQLIMW in the system's background. This paper enforced the concept of authentication by using hash function and a private key along with traditional password. The idea of detection lies in number of queries returned in result set after query execution. In case of, registered user, query set will be not equal to 0 and since username is unique, there will be only one record matching that username, if it exceeds 1, then SQLIMW detects if it's a SQL injection attack or not.

Advantages

- Generation of hash function by XORing password and generated private key substantially reduces computation time along with providing flexibility and scalability to the system

Disadvantages

- The proposed method is ideal for single sign-on system only and its efficiency needs to be tested on other layers.

C. "Runtime Monitors for Tautology based SQL Injection Attacks" (Ramya Dharam and Sajjan G. Shiva)-(2012)

[4]Ramya Dharam and Sajjan G. Shiva proposed a framework to handle tautology based SQLIAs in Java applications using post-deployment monitoring technique. This paper extends the framework they have explained in [5].

The basic idea behind proposed framework is that

- Identify critical variables used for accepting inputs from external environment.
- Identify all valid paths that could be traversed by these critical variables.
- Monitor the behaviour of application during execution with respect to the identified critical variable and its valid paths to detect tautology based SQLIAs.
- When a critical variable violates the checkpoint and traverses an invalid path, the runtime monitor detects the abnormal behaviour and administrator is informed.

Disadvantage

- It works for tautology based SQL Injection attacks only.
- Critical Path identification to reduce complexity can be tricky and could led to false positives.

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

[6]Indrani Balasundram and E.Ramaraj proposed an authentication scheme. The algorithm uses both Advance Encryption Standard (AES) and Rivest-Shamir-Adleman(RSA) to prevent SQL injection attacks. A unique secret key is issued to every user and server uses combination of private key and public key for RSA encryption. In this method, two level of encryption is applied on login query:

- To encrypt user name and password, symmetric key encryption via user's secret key.
- To encrypt the query, asymmetric key encryption via server's public key.

The proposed scheme declares to be very efficient as it requires 961.88ms for encryption or decryption.

Disadvantages

- It doesn't work for URL based SQL injection attacks.
- PKI needs to be maintained and secured.
- Lack of secure registration phase.

E. "TransSQL: A Translation and Validation-based Solution for SQL-Injection Attacks" (Kai Xiang Zhang, Chia-Jun lin et al)-(2011)

[7]Kai Xiang Zhang and et al have suggested a solution in form of TransSQL to detect malicious SQL queries. The proposed scheme duplicates the database into a LDAP database and queries generated by web application is also converted into LDAP queries. These LDAP queries are executed in LDAP database and result is compared with corresponding result from SQL database. In case of mismatch, SQL injection is detected. After detecting a SQL injection request, the result from SQL database is replaced with a null result and returned to the web application.

Advantages
- It is a server side solution, so modification or update is not required in the legacy web application.
- Automatic and platform independent deployment and installation.
- No need to learn LDAP, thus user's training cost is null.

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

[8]Allen Pomeroy and Qing Tan have 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 analyse 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.

Disadvantages
• Not efficient in case of high volume traffic.
• Packet fragmentation attack could bypass detection.

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

[9]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.

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

[10]Jeom-Goo Kim presents an effective approach to detect SQL injection by removing attribute value of queries and then implementing combination of static and dynamic analysis. This approach profile the SQL query generated from normal users and compare this with SQL query generated dynamically by the attacker. It utilizes a function which has the capability to compare the attribute values of static and runtime SQL query in web application. Then by referring symbol table created, logically XORing of fixed and dynamic query after attribute removal declares if the query is normal or not.

Disadvantages
• Developer learning is required.
• Source code adjustment is needed.

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

[11]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.

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

[12]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: 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.
- Verification Phase: After merging, dynamic verifier checks the obfuscated query at atomic level. If no SQL injection found then original query is reconstructed from obfuscated query before delivering it to the database.

Advantages
- No knowledge about obfuscation/deobfuscation technique is required.
- Negligible runtime overhead.
- Platform independent approach.

Table: 3 Comparative Analysis

S.no	Approach- Author	Description
1	Rahul Johari et al	A query model generator based on hybrid approach for PHP applications. Provides cent percent detection for known attacks
2	Gia Jiao et al	Enforcing authentication by introducing a middleware i.e. SQLIMW in the middle. It implements hash mechanism which is faster than any other encryption.
3	Ramya Dharam et al	Injection detection on basis of identified valid path and critical variables
4	Indrani Balasundram et al	RSA and AES encryption used to enforce login query formation.
5	Kai Xiang et al	Duplicates database and queries into a LDAP database which is platform independent and doesn't propose any change in legacy web applications
6	Allen Pomeroy et al	Network forensics are used to analyse recorded network packets, basis of IDS
7	Jeom-Goo Kim et al	Removes attribute values of queries and then dynamic analysis is done on basis of static profile generated.
8	Bist et al	dynamically extracts the query structures from every SQL query location which are intended by the developer
9	Halder et al	Queries are obfuscated and then verified, if clean, queries are reconstructed for execution
10	Ruse et al	Idea behind this framework is based on creating a specific model that deals with SQL queries automatically.
11	Lambert et al	Based on length mismatch of tokenized original and injection queries
12	Wang et al	Web crawlers to detect hidden hyperlinks or web pages behind login forms.

Conclusion

This paper surveys different SQL injection Detection /Prevention techniques and tools proposed in last decade. These techniques vary from enforcing authentication for sign in web applications to the developed tools proposed to analyze queries before execution for any kind of injection.

References

1. The Open Web Application Security Project, "OWASP TOP 10 Project", http://www.owasp.org/
2. Pankaj Sharma, Rahul Johari, S.S Sharma “Integrated approach to prevent SQL injection attack and reflected cross site scripting attack”, Springer, Oct-Dec 2012 pp: 343-351
3. Gao Jiao, Chang-Ming XU, Jing Msohua “SQLIMW: a new mechanism against SQL-Injection “, IEEE International conference on computer science and service system-(2012)
4. Ramya Dharam, Sajjan. G. Shiva, “Runtime Monitors for Tautology based SQL Injection Attacks”, IEEE, 2012
5. Ramya Dharam, Sajjan. G. Shiva, “A Framework for Development of Runtime Monitors”, International Conference on Computer and Information Sciences (ICCIS), Kuala Lumpur, Malaysia, June 2012.
6. Indrani Balasundaram, E.Ramaraj “An Authentication Scheme for Preventing SQL Injection Attack Using Hybrid Encryption (PSQLIAHBE”(ISSN 1450-216X Vol.53 No.3 (2011),pp.359-368)
7. Kai-Xiang Zhang, Chia-Jun Lin, Shih-Jen Chen, Yan ling Hwang, Hao-Lun Huang, Fu-Hau Hsu “TransSQL: A Translation and Validation-based Solution for SQL-Injection Attacks”, IEEE, 2011
8. Pomeroy, A Qing " Effective SQL Injection Attack Reconstruction Using Network Recording" in Computer and Information Technology (CIT), IEEE, 2011
9. 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)
10. Jeom-Goo Kim, Cheonan " 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
11. 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.
12. Raju Halder and Agostino Cortesi, “Obfuscation-based Analysis of SQL Injection Attacks”. 978-1-4244-7755-5/10/$26.00 ©2010 IEEE