ISSN (Online): 2230-7893
www.IJCEM.org

Survey on SQL Injection attacks and their Countermeasures

Nilesh Khochare[1], Satish Chalurkar[2] ,Santosh Kakade[3] and B.B. Meshramm[4]

[1,2,3] Research Scholar, Computer Department, VJTI,
Mumbai, Maharashtra, India
[1]nileshkhochare@gmail.com, [2]satishchalurkar1@gmail.com , [3]santosh20may@gmail.com

[4]Professor, Computer Department, VJTI,
Mumbai, Maharashtra, India
[4]bbmeshram@vjti.org.in

Abstract
SQL injection is most common methodology employed by a hacker to exploit vulnerabilities in software applications. Vulnerabilities are basically weak links in the software that exposes unauthorized data or information to a user. SQL injection occurs when the user input is incorrectly filtered for embedded SQL statements. The technique is powerful enough not only to expose the information to the user but also modify and delete the content which could prove disastrous to the company. There are many ways to prevent SQL attacks such as using dynamic SQL, using automated SQL test tools, by escaping user input.

Keywords - *SQL injection, Web Application, Static Analysis, Runtime Monitoring.*

1. Introduction

An SQL Injection is one of the most common and most dangerous security issues. SQL injections are dangerous because they are a door wide open to hackers to enter your system through your Web interface and to do whatever they want i.e. delete tables, modify databases, even get hold of your corporate network. SQL injection attacks take advantage of code that does not filter input that is being entered directly into a form. Susceptible applications are applications that take direct user input and then generate dynamic SQL that is executed via back-end code. Many webpages take input from users, such as search terms, feedback comments or username and password, and use them to build a SQL query which is passed to the database. If these inputs are not validated, then attacker can insert SQL queries and do malicious activities like delete tables, alter tables etc. SQL injections might be common but, they are also easy to prevent.

2. SQL Injection Basics

SQL injection is the vulnerability that results when you give an attacker the ability to influence the Structured Query Language (SQL) queries that an application passes to a back-end database.
Researchers generally divide injection attacks into three categories: First order Attacks, Second order Attacks and Lateral injection.

2.1 First order Attacks.

The first order attacks are basic attacks. It is When UNIONS or Sub query added to the existing statement.

2.2 Second order Attacks.

In the second order attacks attacker insert the malicious code into the application but not activated immediately by the application. There are various attacks classes in the second order attack.

Frequency based Primary Application: Attacks in this class frequently target the other users of the primary application. For example, topmost searched items, latest popular article.

Frequency based Secondary Application: This class includes application that did not initially receive the injected code, but instead process submission from an application and represent this material for statistical review. Attacks within this type targets on the system administrators.

Secondary Support Application: This class includes application used to internally support primary application. Attacks within this class typically target internal application users and attack activation may be accelerated through social engineering vectors.

Cascaded Submission Application: this class includes application that makes use of multiple client submission within single processing statement. Attacks within this class typically utilize SQL code statements to manipulate the search request and consequently target backend database resources.

2.3 Lateral injection

Using Lateral SQL Injection, an attacker can exploit a PL/SQL procedure that does not even take user input. When a variable whose data type is date or number is concatenated into the text of a SQL statement, there still is a risk of injection.

3. Types of SQL Injection attacks

When the attacker finds an input source that can be used to exploit SQL injection attack vulnerability, there are various types of SQL injection attacks techniques that they can employ.

3.1 Tautologies.

A SQL tautology is a statement that is always true. Tautology-based SQL injection attacks are usually used to bypass user authentication or to retrieve unauthorized data by inserting a tautology into a conditional statement. A typical SQL tautology has the form "or <comparison expression>", where the comparison expression uses one or more relational operators to compare operands and generate an always true condition. The general goal of a tautology-based attack is to inject SQL tokens that cause the query's conditional statement to always evaluate the true. For example,

Select * from Employee where EmpName = ' ' or 1=1 -- ' and Password= 'xxxxx'.

The "or 1=1" is the most commonly known tautology.

3.2 Piggy-Backed Queries

In the piggy-backed Query attacker tries to append additional queries to the original query string. On the successful attack the database receives and executes a query string that contains multiple distinct queries. In this method the first query is original whereas the subsequent queries are injected. This attack is very dangerous; attacker can use it to inject virtually any type of SQL command. For example,

SELECT info FROM employee WHERE login-'abc' AND pin-0; drop table employee.

Here database treats above query string as two query separated by ';', and executes both. The second sub query is malicious query and it causes the database to drop the employee table in the database. There are so many other types of queries such as inserting new employees in the database.

3.3 Logically Incorrect Queries

This attack takes advantage of the error messages that are returned by the database for an incorrect query. These database error messages often contain useful information that allow attacker to find out the vulnerable parameter in an application and the database schema. Suppose after inserting a incorrect query if attacker gets following error message,

"Microsoft OLEDB provider for SQL Server (0×80040E07)Error converting nvarchar value 'CreditCards' to a column of data type int"

Now there are two useful pieces of information in this error message. First, the attacker come to know that the database in in SQL server database. Second, attacker comes to know that the name of the first user-defined table in the database is "CreditCards". So by using the same strategy attacker can find the name and type of each column in the given table.

3.4 Union Query

Union query injection is called as statement injection attack. In this attack attacker insert additional statement into the original SQL statement. This attack can be done by inserting either a UNION query or a statement of the form ";< SQL statement >" into vulnerable parameter. The output of this attack is that the database returns a dataset that is the union of the results of the original query with the results of the injected query. For example,

"Select * from users where UserName=' ' union select * from employee –'and Password='anypwd' "

The above query becomes the union of two SELECT queries. Here first query returns a null set because of no matching records in the table USERS. The second query returns all the data from the table EMPLOYEE.

3.5 Stored Procedure

In this technique, attacker focuses on the stored procedures which are present in the database system. Stored procedures run directly by the database engine. Stored procedure is nothing but a code and it can be vulnerable as program code. [8] For authorized/unauthorized user the stored procedure returns true/false. As an SQLIA, intruder input " , ; SHUTDOWN; - -" for username or password. Then the stored procedure generates the following query:

SELECT accounts FROM users WHERE login= 'doe' AND pass=' '; SHUTDOWN; -- AND pin =

This type of attack works as piggy-back attack. The first original query is executed and consequently the second query which is illegitimate is executed and causes database shut down. So, it is considerable that stored procedures are as vulnerable as web application code.

3.6 Inference

This type of attack create queries that cause an application or database to behave differently based on the result of the query. These attacks allow an attacker to extract data from the database and detect vulnerable parameter. There are to well-known attack techniques based on inference: blind-injection and timing attacks.

Blind-injection
An attacker performs queries that have a Boolean result. If the answer is true then the application behaves correctly and if the answer is false then it cause an error. So attacker can get the indirect response from database.

Timing attacks
In this attack attacker observe the database delays in the database response and gather the information. To perform the timing attack attacker writes the query in the form of an if-then statement and then uses the WAITFOR keyword in one of the branch, which causes the database to delay its response by specified time.

3.7 Alternate Encodings

To avoid the signature and filter based checks the attacker modify their injection strings called as alternate encoding technique, such as ASCII, Hexadecimal and Unicode can be used in conjunction with other techniques to allow an attack and to escape from various detection methods.

4 SQL injection prevention tools

There are many ways to prevent SQL injection attacks. [6] The most popular in the source code. There are some approaches for testing Web applications to identify the presence of SQL injection vulnerabilities, e.g. using black-box testing techniques.methods are tainting and tracking of the user input, analyze the correctness of SQL statement statically; appending random numbers to SQL statements

4.1 VIPER tool for penetration testing.

[4]According to Angelo Ciampa, Corrado Aaron Visaggio and Massimiliano Di Penta, they have suggested a tool called Viper to perform penetration testing of Web applications. This tool relies on a knowledge base of heuristics that guides the generation of the SQL queries. This tool first identifies the hyperlink structure and its input form.

4.2 Attack Injection Methodology.

[7]Joao Antunes,Nuno Neves,Miguel Correia, Paulo Verissimo and Rui Neves has suggested the attack injection methodology i.e.AJECT tool which adapts and extends classical fault injection techniques to look for security vulnerabilities. In this Attack Injection Tool first the attacks are generated on the target system to evaluate the system. Means they first build test cases that would not only exercise all reachable computer instructions but also try them with every possible instance of input.

4.3 Static analysis And Runtime Monitoring Tool.

[9]William G.J. Halfond and Alessandro Orso has suggested the tool that detects and prevents SQL injection attacks by combining static analysis and runtime monitoring. The name of the tool is AMNESIA (Analysis and Monitoring for NEutralizing SQL-Injection Attacks). This tool uses both static and runtime analysis. At static analysis it analyse the web application code and at runtime this techniques monitors all the dynamically generated queries.

4.4 Identifying SQL and XSS vulnerability.

[11]Adam Kie˙zun, Philip J. Guo, Karthick Jayaraman, Michael D. Ernst has suggested a technique for finding vulnerabilities in Web Application such as SQL attack and Cross site scripting(XSS).This technique works on existing code, creates concrete inputs that expose vulnerabilities and operates before software is deployed. It analyses application internals to discover vulnerable code. The tool that identifies the SQL and XSS vulnerability is known as ARDILLA. It is based on input generation, taint propagation, and input mutation to find variants of an execution that exploit vulnerability.

4.5 Obfuscation-based Analysis of SQL Injection Attacks

[13]Raju Halder and Agostino Cortesi proposes the obfuscation/deobfuscation based based technique to detect the presence of possible SQL Injection Attacks (SQLIA) in a query before submitting it to a DBMS. Now the Obfuscated code is a source code that has been made difficult for human. [13]In obfuscation approach the possible attack injection are verified at atomic formula level and only those atomic formulas which are tagged as vulnerable, also this approach avoids the root cause of SQL injection attacks in dynamic query generation .

4.6 SQLInjectionGen SQLIA Detector

[12] MeiJunjin has suggested a tool SQLInjectionGen tool which combines the static analysis, runtime analysis and automatic testing. This is an automated test case generation tool to identify SQL injection vulnerability. According to author the prototype tool SQLInjectionGen had no false positives and small number of false negatives.

4.7 SQLrand Practical Protection mechanism

[21] S. W. Boyd and A. D. Keromytis has suggested the practical protection mechanism for preventing SQL injection attacks against web server. This tool uses SQL randomized query CGI application and detect and correct the queries injected into the code.

4.8 CANDID: Dynamic candidate Evaluations

P. Bisht, P. Madhusudan, and V. N.Venkatakrishnan has suggested dynamic candidate evaluation approach for automatic prevention of SQL injection attacks. This tool dynamically extracts the query structures from every SQL query location which are intended by the programmer.

5 Conclusion

SQL injection is most powerful and easiest attack method on the Web Application. In this paper we have studied many SQL attack prevention methods proposed by various authors. In this the VIPER tool performs the penetration testing by using the standard SQL injections. [6]VIPER tool successfully discovers the SQL vulnerabilities within the Web Application. Whereas the ADJECT tool first generate the attack on the target system to evaluate the system and build the test cases.[7]The AJECT tool could detect different classes of vulnerabilities in e-mail servers and assist the developers in their removal by providing the required test cases. The AMNESIA tool uses both static analysis and runtime monitoring.[9]AMNESIA is a fully automated tool for protecting Web applications against SQL injection attacks. [11]The ARDILLA technique is based on input generation, dynamic taint propagation, and input mutation to find a variant of the input that exposes vulnerability. The ARDILLA tool finds both SQL and XSS vulnerabilities

References

[1] Frank S. Rietta:"Application Layer Intrusion Detection for SQL Injection", ACM SE'06 March 10-12, 2006, Melbourne, Florida, USA.

[2] Ryan Riley, Xuxian Jiang, and Dongyan Xu:"An Architectural Approach to Preventing Code Injection Attacks", IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 7, NO. 4, OCTOBER-DECEMBER 2010.

[3] Elias Levy, Iván Arce: "New Threats and Attacks on the World Wide Web", PUBLISHED BY THE IEEE COMPUTER SOCIETY, 1540-7993/06/$20.00 © 2006 IEEE,IEEE SECURITY & PRIVACY.

[4] Angelo Ciampa, Corrado Aaron Visaggio, Massimiliano Di Penta :"A heuristic-based approach for detecting SQL-injection vulnerabilities in Web applications".

[5] Joa˜o Antunes,Nuno Neves, Miguel Correia, Paulo Verissimo and Rui Neves: "Vulnerability Discovery with Attack Injection", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 36, NO. 3, MAY/JUNE 2010

[6] William G.J. Halfond and Alessandro Orso: "Preventing SQL Injection Attacks Using AMNESIA", ICSE'06, May 20–28, 2006, Shanghai, China.ACM 1-59593-085-X/06/0005.

[7] Adam Kie˙zun,Philip J. Guo,Karthick Jayaraman,Michael D. Ernst:"Automatic Creation of SQL Injection and Cross-Site Scripting AttackS", ICSE'09, May 16-24, 2009, Vancouver, Canada,978-1-4244-3452-7/09/$25.00 © 2009 IEEE.

[8] Atefeh Tajpour, Maslin Masrom, Mohammad Zaman Heydari, Suhaimi Ibrahim: "SQL Injection Detection and Prevention Tools Assessment." 978-1-4244-5540-9/10/$26.00 ©2010 IEEE

[9] Diallo Abdoulaye Kindy and Al-Sakib Khan Pathan: "A survey on SQL injection: vulnerabilities, attacks, and prevention techniques". 2011 IEEE 15th International Symposium on Consumer Electronics.

[10] Cristian Pinzón, Juan F. De Paz, Javier Bajo, Álvaro Herrero, Emilio Corchado:"AIIDA-SQL: An Adaptive Intelligent Intrusion Detector Agent for Detecting SQL Injection Attacks". 2010 10th International Conference on Hybrid Intelligent Systems.

[11] Li Shan, Dong Xiaorui, RaoHong:"An Adaptive Method Preventing Database from SQL Injection Attacks". 2010 3rd International Conference on Advanced Computer Theory and Engineering (1CACTE).

[12] MeiJunjin: "An approach for SQL injection vulnerability detection". 2009 Sixth International Conference on Information Technology: New Generations.

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

[14] Jan-Min Chen, Chia-Lun Wu: "An Automated Vulnerability Scanner for Injection Attack Based on Injection Point". 978-1-4244-7640-4/10/$26.00 ©2010 IEEE.

[15] Michelle Ruse, Tanmoy Sarkar, Samik Basu: "Analysis & Detection of SQL Injection Vulnerabilities via Automatic Test Case Generation of Programs". 2010 10th Annual International Symposium on Applications and the Internet.

[16] Jie Wang, Raphael C.-W. Phan, John N. Whitley and David J. Parish: "Augmented Attack Tree Modeling of SQL Injection Attacks". 978-1-4244-5265-1/10/$26.00 ©2010 IEEE

[17] Atefeh Tajpour, Maslin Massrum, Mohammad Zaman Heydari:"Comparison of SQL Injection Detection and Prevention Techniques". 201O 2nd International Conforence on Education Technology and Computer (ICETC).

[18] Atefeh Tajpour, Mohammad JorJor zade Shooshtari: "Evaluation of SQL Injection Detection and Prevention Techniques". 2010 Second International Conference on Computational Intelligence, Communication Systems and Networks.

[19] NTAGW ABIRA Lambert, KANG Song Lin: "Use of Query Tokenization to detect and prevent SQL Injection Attacks". 978-1-4244-5540-9/10/$26.00 ©2010 IEEE.

[20] Xin Wang, Luhua Wang, Gengyu Wei, Dongmei Zhang, Yixian Yang, "HIDDEN WEB CRAWLING FOR SQL INJECTION DETECTION". 978-1-4244-6769-3/10/$26.00 ©2010 IEEE.

[21] S. W. Boyd and A. D. Keromytis. SQLrand: Preventing SQL Injection Attacks. In Proceedings of the 2nd Applied Cryptography and Network Security Conference, pages 292–302, June 2004.

[22] 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.