International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2017 IJSRCSEIT | Volume 2 | Issue 6 | ISSN : 2456-3307

SQL Pen-Testing framework for Cyber Security : A Review

Ravi Nayak[*1], Dr. Priyanka Sharma[2]
[*1]Research Scholar, MTECH Department, Raksha Shakti University, Ahmedabad, Gujarat, India
[2]Professor, MTECH Department, Raksha Shakti University, Ahmedabad, Gujarat, India

ABSTRACT

In Modern Life, Cyber Security is a major concern nowadays. Web application is not secured and there exits bugs or vulnerability found in a web application. So the major attack in a web application is Injection attack in which SQL injection has a high priority. In this Paper, we proposed Steps for Penetration testing of SQL Injection to measure or to detect attacks such as SQL Manipulation, Code Injection Function Call Injection, Buffer Overflow, Error Based SQL Injection and Blind SQL injection.

Keywords: SQL, Penetration Testing, SQL Injection

I. INTRODUCTION

SQL injection is a code injection technique, used to attack data-driven applications, in which nefarious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker).SQL injection must exploit a security vulnerability in an application's software, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed. SQL injection is mostly known as an attack vector for websites but can be used to attack any type of SQL database.[1]

SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, and become administrators of the database server.In a 2012 study, it was observed that the average web application received 4 attack campaigns per month, and retailers received twice as many attacks as other industries.[1]

The SQL injection cheat sheet is a resource in which you can find detailed technical information about the many different variants of the SQL Injection vulnerability. This cheat sheet is a good reference to both seasoned penetration tester and also those who are just getting started in web application security.[2]

The first step in this test is to understand when the application interacts with a DB Server in order to access some data. Typical examples of cases when an application needs to talk to a DB include:

Authentication forms:

(1) when authentication is performed using a web form, chances are that the user credentials are checked against a database that contains all usernames and passwords (or, better, password hashes).
(2) Search engines: the string submitted by the user could be used in a SQL query that extracts all relevant records from a database.
(3) E-Commerce sites: the products and their characteristics (price, description, availability, etc) are very likely to be stored in a database[3]

Intrusion Detection Based on Static Analysis: Using static string analysis technique,[7] it is possible to construct a regular expression that conservatively approximates the set of SQL statements generated at a hotspot (which submits SQL query). The information can be used to statically analyze syntax correctness of SQL statements [5] and to model “normal behaviors” of a Web application. During run time, any SQL

statement not contained in the approximation library will be rejected by intrusion detection.[6]

Black-box testing can be used to discover SIA vulnerabilities, by applying a library of pre-collected attack patterns. It is fast and effective, however, without prior knowledge of source code, it has difficulty in discovering non-trivial vulnerabilities.[8][9]

II. SQL PEN-TESTING FRAMEWORK

Step 1: List All Input Fields and Hidden Fields of POST Requests

By Making all input fields and hidden fields of POST requests whose values could be used in crafting a SQL query.

Step 2: **Perform Information Gathering**

By understanding the web application architecture and the technologies used and to Determine the DB name, version, users, output mechanism, DB type, user Privilege level, and OS interaction level, also Check error message for database information.

Step 3: Attempt to inject codes into the Input Fields to Generate an Error.

(1) Attempt to Inject codes into the input fields to generate an error: a single quote(‘), a semicolon(;), comments(-), AND, and OR
(2) Try to insert a string value where a number is expected in the input field

Step 4: Try to find SQL Injection Vulnerabilities by Interface, Manipulating a Parameter, Using Database Errors and Application Response

By Understanding how a web browser sends requests to the web server, also Observe the GET requests and POST requests parameters and Use the Burpsuite tool to modify the data

Step 5: Perform Fuzz Testing, Function Testing, Static and Dynamic testing to detect SQL vulnerabilities

By using Fuzzing Tool to find out SQL Vulnerabilities in the Application and to fix semantic bugs in the code, also try to insert a string value where a number is expected or vice versa to generate an error.

Examples of function testing :

http://www.website.com?parameter=123
http://www.website.com?parameter=1’
http://www.website.com?parameter=1”
http://www.website.com?parameter=1 AND 1=1-

Step 6: Perform Error based Injection Attack

- ✓ Check whether the application’s database error messages are disclosed to users.
- ✓ Try to build vulnerability exploit query requests from database error messages
- ✓ Use HP WebInspect to detect potential Error-based SQL injection.

Step 7: Perform Blind SQL Injection Attack

The Blind SQL Injection is performed when an error message is not received from the application while trying to exploit SQL vulnerabilities. The developer’s specific message is displayed instead of an error message.SO it is quite difficult to find SQL vulnerability in such cases. Also, Try to enumerate First Table Entry and Extract Data from Rows using Blind SQL injection

Step 8: Determine Privileges, DB Structure, and Column Names

(1) Identify User Level Privilege
(2) DB Administrators
(3) Discover DB Structure
(4) Column Enumeration in DB

III. RESULTS

As a Result, we can use manual testing approach for Attacks such as SQL Manipulation, Code Injection Function Call Injection, Buffer Overflow, Error Based SQL Injection and Blind SQL injection.

IV. CONCLUSION AND FUTURE WORK

We concluded that this paper proposed and outlined the steps of Penetration testing for SQL Injection and In future, we will use this steps for finding and detecting SQL Injection attacks such as SQL Manipulation, Code Injection Function Call Injection, Buffer Overflow, Error Based SQL Injection and Blind SQL injection.

V. REFERENCES

[1]. https://en.wikipedia.org/wiki/SQL_injection

[2]. https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/

[3]. https://www.owasp.org/index.php/Testing_for_SQL_Injection_(OTG-INPVAL-005)

[4]. A. Christensen, A. Møller, and M. Schwartzbach. Precise analysis of string expressions. In Proceedings of the International Static Analysis Symposium (SAS'03), 2003

[5]. C. Gould, Z. Su, and P. Devanbu. JDBC Checker: A Static Analysis Tool for SQL/JDBC Applications. In Proceedings of the 26th International Conference on Software Engineering, pages 697-698, 2004.

[6]. W. Halfond and A. Orso. AMNESIA: Analysis and Monitoring for NEutralizing SQL-Injection Attacks. In Proceedings of the 20th IEEE/ACM international Conference on Automated software enginee, pages 174-183, 2005.

[7]. J. Burch, E. Clarke, K. McMillan, D. Dill, and L. Hwang. Symbolic model checking: 1020 states and beyond. In IEEE Symposium on Logic in Computer Science, pages 428-439, 1990.

[8]. SPI Dynamics. Webinspect: Security throughout the application lifecycle. SPI Dynamics.Datasheet. http://www.spidynamics.com/assets/documents/WebInspect_DataSheets.pdf

[9]. Y.W. Huang, S.K. Huang, T.P. Lin, and C.H. Tsai.Web application security assessment by fault injection and behavior monitoring. In Proceedings of the 11th International World Wide Web Conference (WWW 2003), 2003.