April 2013

Analysis of SQL Injection Attack

Jayeeta Majumder

*Department of Computer Science and Engineering, Haldia Institute of Technology , Haldia, West Bengal
India, jem2003_kolkata@yahoo.co.in*

Gargi Saha

*Department of Computer Science and Engineering, Haldia Institute of Technology , Haldia, West Bengal
India, gargisaha88@yahoo.co.in*

Follow this and additional works at: <https://www.interscience.in/ijcsi>

Part of the [Computer Engineering Commons](#), [Information Security Commons](#), and the [Systems and Communications Commons](#)

Recommended Citation

Majumder, Jayeeta and Saha, Gargi (2013) "Analysis of SQL Injection Attack," *International Journal of Computer Science and Informatics*: Vol. 2 : Iss. 4 , Article 7.

DOI: 10.47893/IJCSI.2013.1102

Available at: <https://www.interscience.in/ijcsi/vol2/iss4/7>

This Article is brought to you for free and open access by the Interscience Journals at Interscience Research Network. It has been accepted for inclusion in International Journal of Computer Science and Informatics by an authorized editor of Interscience Research Network. For more information, please contact sritampatnaik@gmail.com.

Analysis of SQL Injection Attack

Jayeeta Majumder & Gargi Saha

Department of Computer Science and Engineering, Haldia Institute of Technology, Haldia, West Bengal India
E-mail : jem2003_kolkata@yahoo.co.in, gargisaha88@yahoo.co.in

Abstract - SQL injection attacks are a serious security threat to Web applications. They allow attackers to obtain unrestricted access to the databases underlying the applications and to the potentially sensitive information these database contain. Various researchers and practitioners have proposed various methods to address the SQL injection problem. To address this problem, we present an extensive review of the various types of SQL injection attacks known to date. For each type of attack, we provide descriptions and examples of how attacks of that type could be performed. We also present a methodology to prevent SQL injection attacks. It concentrates on the SQL queries and SQL stored procedure where input parameters are injected by the attacker. After a rigorous input validation with our proposed SQL security model will ensure input validation.

Keywords - SQL Injection, Attack Intent, SQLIA Detector, VIPER tool.

I. INTRODUCTION

As the Internet is growing day by day, most of the people are not aware of security and privacy. Internet is a widespread information infrastructure. It is basically an insecure channel for exchanging information. Web security is the set of rules and measures taken against web security threats. Most of the applications have the vulnerability that makes a threat possible. An attack may be possible due to poor design, configuration mistakes, or poor written code of the web application. All types of code injection or SQL injection are very dangerous for these components of the web application. The number of SQLIA's [6,16] reported in the past few years has been showing a steadily increasing trend. It is, therefore, of huge importance to prevent such types of attacks, and SQLIA prevention has become one of the most active topics of research in the industry and academia. There has been significant progress in the field and a number of models have been proposed and developed to counter SQLIA's, but none have been able to guarantee an absolute level of security in web applications, mainly due to the diversity and scope of SQLIA's. SQL injection refers to a class of code-injection attacks in which data provided by the user is included in an SQL query in such a way that part of the user's input is treated as SQL code. By leveraging these vulnerabilities, an attacker can submit SQL commands directly to the database. These attacks are a serious threat to any Web application that receives input from users and incorporates it into SQL queries to an underlying database. Most Web applications used on the Internet or within enterprise systems work this way and could

therefore be vulnerable to SQL injection. The cause of SQL injection vulnerabilities is relatively simple and well understood: insufficient validation of user input. In addition, even if the injected code is intercepted before execution, administrators are often presented with information that does not identify clearly the association between the commands that were attempted, the assets that were at risk, the threats that were imposed, and the countermeasures he/she has at disposal. To address these issues, a repository of SQL injection attacks that are classified in a semantic-aware, easy to comprehend model is needed [7].

II. COMPUTER SYSTEM AND AN EXAMPLE OF SQL INJECTION

SQL injection is a vulnerability that allows an attacker to influence the queries that are passed to the back-end database.

Three Tier Architecture :

Simple three tier database-driven architecture [6,15] have three layers

- Presentation Tier (Browsers)
- Logic Tier
- Storage Tier (Database)

The three tier architecture maintains linear relationship i.e. Presentation Tier connects to the Logic Tier and the Logic Tier connects to the Storage Tier

Presentation Tier <-> Logic Tier <-> Storage Tier

Four Tier Architecture :

In Four Tier Architecture [6] an layer of Application Server is inserted between a web server and the database.

- Presentation Tier (Browsers)
- Logic Tier
- Application Tier
- Storage Tier (Database)

Fig. 1: Four Tier architecture database driven architecture

In four tier architecture the connection to the database is opened by application server which has Application Programming Interface (API) that implements the business logic before transferring the data to the Logic tier.

III. WORKING OF SQL INJECTION

SQL injection can be used [6] using various methods. In this tutorial we will explain to the basic concepts behind the SQL injection. Suppose you are on a shopping site and you have selected the option of showing all the accessories that costless then 200\$ and its URL is like

`http://www.shoppingsite.com/products.php?val=100`

To test this website for SQL injection try appending your SQL injection commands in the Val parameter 'OR '1'='1'

`http://www.shoppingsite.com/products.php?val=100'OR '1'='1`

If the above injection works and shows the list of all the accessories then the website is vulnerable to this type of SQL injection

This means that at the backend the script will be executed as shown:

```
SELECT * FROM Products WHERE Price<'200.00'
OR '1'='1' ORDER BY Product description
```

As the condition 1=1 so this will give you list of all the products.

- How this SQL injection Attack Can Be Harmful:

Suppose a website uses the following url for logging into admin panel

`http://www.website.com/cms/login.php?username=saini&password=go`

Now if the above website is vulnerable to the SQL injection as mentioned in the above example then by entering any username and password in the URL you can login

`http://www.website.com/ms/login.php?username=dnt&password=dnt'OR '1'='1`

So you will just login without valid username and password to the admin panel of a website.

This is becoming very long so I will explain the rest and the best methods of SQL injection in my next post On SQL injection.

IV. ATTACK INTENT

Attacks can also be characterized based on the goal, or intent, of the attacker. Therefore, each of the attack type definitions [13] that

Identifying injectable parameters:

The attacker wants to probe a Web application to discover which parameters and user-input fields are vulnerable to SQLIA.

Performing database finger-printing:

The attacker wants to discover the type and version of database that a Web application is using. Certain types of databases respond differently to different queries and attacks, and this information can be used to "fingerprint" the database. Knowing the type and version of the database used by a Web application allows an attacker to craft database specific attacks.

Determining database schema:

To correctly extract data from a database, the attacker often needs to know database schema information, such as table names, column names, and column data types. Attacks with this intent are created to collect or infer this kind of information.

Extracting data:

These types of attacks employ techniques that will extract data values from the database. Depending on the type of the Web application, this information could be sensitive and highly desirable to the attacker. Attacks with this intent are the most common type of SQLIA.

V. TECHNIQUES OF SQLIA'S

Most of the attacks are not in isolated they are used together or sequentially, depending on the specific goals of the attacker.

Tautologies :

Tautology-based attack [4] is to inject code in one or more conditional statements so that they always evaluate to true. The most common usages of this technique are to bypass authentication pages and extract data. If the attack is successful when the code either displays all of the returned records or performs some action if at least one record is returned.

Example: In this example attack, an attacker submits “ ‘ or 1=1 - - ”

The Query for Login mode is: `SELECT * FROM user_info WHERE loginID=’ or 1=1 - - AND pass1=’`

The code injected in the conditional (OR 1=1) transforms the entire WHERE clause into a tautology the query evaluates to true for each row in the table and returns all of them. In our example, the returned set evaluates to a not null value, which causes the application to conclude that the user authentication was successful. Therefore, the application would invoke method `user_main.aspx` and to access the application.

Union Query :

In union-query attacks [5], Attackers do this by injecting a statement of the form: UNION

`SELECT <rest of injected query>` because the attackers completely control the second/injected query they can use that query to retrieve information from a specified table. The result of this attack is that the database returns a dataset that is the union of the results of the original first query and the results of the injected second query.

Example: An attacker could inject the text “ ‘ UNION SELECT pass1 from user_info where LoginID=’secret - - ” into the login field, which produces the following query: `SELECT pass1 FROM user_info WHERE loginID=’`

`UNION SELECT pass1 from user_info where LoginID=’secret’ -- AND pass1=’`

Assuming that there is no login equal to “”, the original first query returns the null set, whereas the second query returns data from the “user_info” table. In this case, the database would return column “pass1” for account “secret”. The database takes the results of these two queries, unions them, and returns them to the application.

In many applications, the effect of this operation is that the value for “pass1” is displayed along with the account information.

VI. SQL INJECTION PREVENTION TOOLS

There are many ways to prevent SQL injection attacks. 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

VIPER tool for Penetration Testing :

According to Angelo Ciampa, Corrado Aaron Visaggio and Massimiliano Di Penta, they have suggested a tool called Viper [3] 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. Angelo Ciampa, Corrado Aaron Visaggio, Massimiliano Di Penta :”A heuristic-based approach for detecting SQL-injection vulnerabilities in Web applications”.

Attack Injection Methodology :

Joao Antunes,Nuno Neves,Miguel Correia, Paulo Verissimo and Rui Neves has suggested the attack injection methodology i.e.AJECT tool [2] 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.

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/ © 2009 IEEE

SQL InjectionGen SQLIA Detector :

MeiJunjin has suggested a tool SQLInjectionGen tool [1] 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. MeiJunjin: "An approach for SQL injection vulnerability detection". 2009 Sixth International Conference on Information Technology: New Generations.

VII. PROPOSED SOLUTION

It is not compulsory for an attacker to visit the web pages using a browser to find if SQL injection is possible on the site. Generally attackers build a web crawler to collect all URLs available on each and every web page of the site. Web crawler is also used to insert illegal characters into the query string of a URL and check for any error result sent by the server. If the server sends any error message as a result, it is a strong positive indication that the illegal special meta character will pass as a part of the SQL query, and hence the site is open to SQL Injection attack. For example Microsoft Internet Information Server by default shows an ODBC error message if an any meta character or an unescaped single quote is passed to SQL Server. The Web crawler only searches the response text for the ODBC messages.

Step1: Indexing Phase

Identify which user input are used in SQL Query or Stored Procedure in Application. Dependent SQL Query or Stored Procedure are given a unique Number.

Step2 : Checking with our Security Model

Now in Second Phase it takes input and check with a array structure or lookup table for Inputs validation for Data Type, Length and White Listed SQL Keywords and Limited Special Character.

Step3 : Reencoding of ASCII Equivalent of Characters

Now Convert all Input parameter characters into its ASCII equivalent then Reconstruct the same from this ASCII equivalent.

Then if these three validation passed then process it for further execution Else generate an SQLIA Event and log it for further analysis.

In the application there are SQL Text Query and Stored Procedures. Our intention is to identify the asset area and vulnerable portions in which attacker make targets. Only the input parameters Which are used to build the SQL Text Query and Stored Procedure are taken as spots where we place Token id to identify the SQL Text Query and Stored Procedure of the selected query.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we have presented a survey of current techniques of SQL injection as well as a solution methodology for preventing SQLIAs. To perform this evaluation, I first identified the various types of SQLIAs known to date. we also studied the different mechanisms through which SQLIAs can be introduced into an application and identified which techniques were able to handle which mechanisms. Many of the techniques have problems handling attacks that take advantage of poorly-coded stored procedures and SQL queries cannot handle attacks. This difference could be explained by the fact

that prevention-focused techniques try to incorporate defensive coding best practices into their attack prevention mechanisms.

Future work should focus on evaluating the techniques precision and effectiveness in practice. Empirical evaluations will be performed which allow comparing the performance of the different techniques when they are subjected to real-world attacks and legitimate inputs. As well as precision, accuracy and add on will be incorporated for the prevention of much complex SQLIA such as Linked Server, Internal Network Attack etc.

REFERENCES

- [1] MeiJunjin: "An approach for SQL injection vulnerability detection". 2009 Sixth International Conference on Information Technology: New Generations.
- [2] Joao Antunes, Nuno Neves, Miguel Correia, Paulo Verissimo and Rui Neves has suggested the attack injection methodology in their paper named "Using Attack Injection to Discover New Vulnerabilities"
- [3] Angelo Ciampa, Corrado Aaron Visaggio, Massimiliano Di Penta : "A heuristic-based approach for detecting SQL-injection vulnerabilities in Web applications".
- [4] OWASP The open web application security project https://www.owasp.org/index.php/SQL_Injection_Cookbook_-_Oracle
- [5] SQL Injection Cheat Sheet <http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/>
- [6] Basics And Working of SQL Injection Attacks <http://www.pctipstricks.net/hacking/basic-working-sql-injection/>
- [7] <http://www.phrack.org/>
- [8] Stephen Thomas and Laurie Williams, "Using Automated Fix Generation to Secure SQL Statements" Proc.IEEE CNF, 2007
- [9] Raju Halder and Agostino Cortesi, "Obfuscation-based Analysis of SQL Injection Attacks". 978-1-4244-7755-5
- [10] Black Hat Briefings USA ,Cameron Hotchkies,"Blind SQL Injection Automation Techniques",2004.
- [11] Gregory T. Buehrer, Bruce W. Weide, and Paolo A. G. Sivilotti,"Using Parse Tree Validation to Prevent SQL Injection Attacks", 2005
- [12] Ofer Maor and Amichai Shulman, "Blindfolded SQL injection."
- [13] Halfond, viegas, orso "A Classification of SQL Injection Attacks and Countermeasures", ISSSE , 2006
- [14] Chip Andrews, "SQL Injection FAQ" on the 23rd of October at SQLSecurity.com
- [15] Chris Anley, "Advanced SQL Injection in SQL Server Applications.", An NGSSoftware Insight Security Research (NISR) Publication, 2002.
- [16] <http://www.sqlsecurity.com/>