INDIAN INSTITUTE OF INFORMATION TECHNOLOGY

Indian Institute of Information Technology, Dharwad

Final Report

SQL Injection Attack and its Prevention

Sai Deepak Reddy Emani

17BCS11

Supervised by

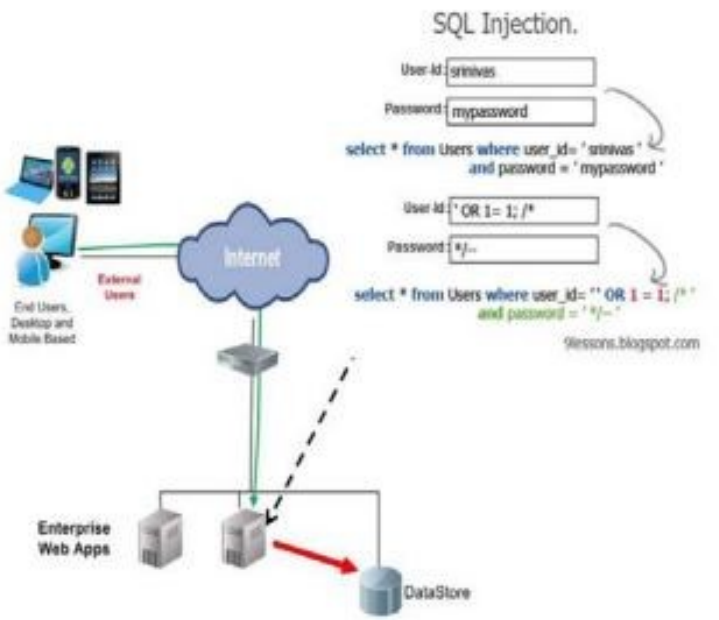
Dr. Uma S

23rd October, 2019

Introduction

- Scope:

  There are many web attacks hacker follows to attack on your web servers. SQL Injection is mostly used attack mechanisms used by hackers to steal data from web server or manipulate them. It is surely one of the most common known application layer attack techniques used in present time. It is the type of attack which takes advantage of improper coding of your web applications that consequently allows hacker to inject SQL commands into a login form and allows them to gain access to the data held within the database. SQL Injection targets the web servers and web applications that use a back-end database.



SQL injection is a very important to study now days, here are some interesting facts for a few different reasons in SQL injection: -

- It’s getting increasingly tougher to write vulnerable codes due to frameworks that automatically parameterize inputs while designing web applications yet we still write bad code.
- It is very easy to detect remotely by automated tools which can be orchestrated to crawl the web searching for vulnerable websites yet you are still putting them out there.

Literature Study:

- SQL Injection allows a user specified query to execute in the database
- Many web applications take user input from a form
- Often this user input is used literally in the construction of a SQL query submitted to a database. For example:
    - SELECT productdata FROM table WHERE productname = '*user input product name*';
- Since the data we're filling in appears to be in the WHERE clause, let's change the nature of that clause *in an SQL legal way* and see what happens. By entering anything' OR 'x'='x, the resulting SQL is:
- As the 'x'='x' clause is guaranteed to be true no matter what the first clause is, hence returns a message
- The credentials has been mailed to random.person@site.com
- Though we haven't got what we expected, but we know that we're able to manipulate the query to our own ends

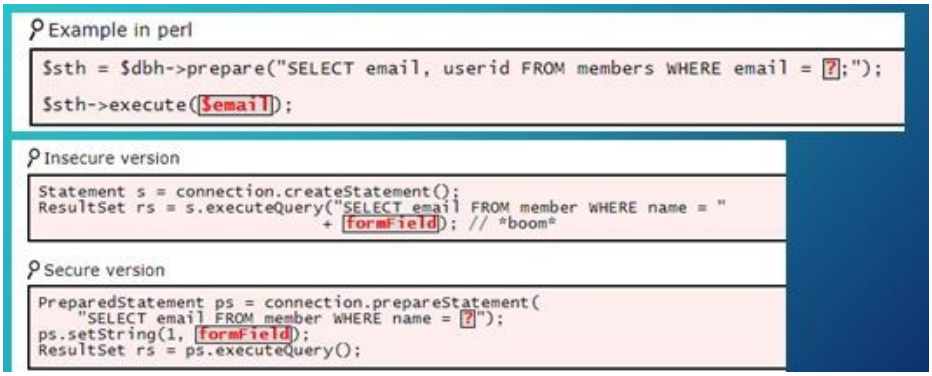
Using Brute force password guessing in order to find the credentials.

- A guessing work performed where there is less protection.
- We'll instead do actual password testing in our snippet by including the email name and password directly. In our example, we'll use our victim, bob@example.com and try multiple passwords.
- This is clearly well-formed SQL, so we don't expect to see any server errors, and we'll know we found the password when we receive the "your password has been mailed to you" message.

Mitigations and SQL Preventions.

1) **Sanitize the input**
    - Strip out the 'bad stuff'- quotes, semicolons or escapes
    - In a precise way remove everything, but known good data.
    - Reject the characters that could not be valid, presumably with an error message- even eliminate typos

2) **Use Bound Parameters**
    - An SQL statement string is created with placeholders - a question mark for each parameter - and it's compiled ("prepared", in SQL parlance) into an internal form.



Types of SQL Injection Attacks:

- Tautology- Which is always true.(1=1)
- Logically incorrect queries. – Giving wrong input.
- Union Query-- The attacker provides the incorrect data with the few correct fields, the SQL query is sent with the 'Union' of both correct and incorrect fields
- SELECT acc_inf FROM clients WHERE login='' UNION SELECT card_No FROM Credit_Cards WHERE acct_No=12450 -- AND pass='' AND pin=
- **Piggy-backed queries** is a type of attack that compromises a database using a query delimiter, such as ";", to inject additional query statements to the original query. In this method the first query is original whereas the subsequent queries are injected.
- SELECT acc_No FROM client WHERE login = 'ritu' AND pass = ''; DROP TABLE client -- 'AND pin = 321

Proposed Solutions - Research Papers

1) SQL Injection Impact on Web Server and Their Risk Mitigation Policy Implementation Techniques: An Ultimate solution to Prevent Computer Network from Illegal Intrusion, Volume 8, No. 3, March – April 2017, International Journal of Advanced Research in Computer Science

This research paper proposed the dynamic technique:

1.1) **AIIDA-SQL:** This method suggests a hybrid approach based on Adaptive Intelligent Intrusion Detection (AIIDA-SQL) for detection of SQLIA. AIIDA-SQL combines the advantages of Case Based Reasoning (CBR) systems, such as learning and adaptation, with the predictive capabilities of a combination of Artificial Neural Network (ANN) and Support Vector Machine (SVM). Through these mechanisms, their take advantages of both strategies in order to trustworthy classifying the SQL queries. In final manner, in order to classify SQL queries as distrustful, utilized a virtualization mechanism, which combines clustering techniques and unsupervised neural models to reduce dimensionality of the data.

1.2) A query tokenization based method:

This is a technique based query tokenization is proposed for detect and prevent SQL injection attacks. This method checks user inputs whether their cause changes query's intended result. At the next step, this method tokenizing original query and malicious injected query, separately. After tokenizing, two arrays are created by all tokens. Finally, the lengths of obtained arrays are compared. If their length be different, an injection attack is detected.

1.3) A learning based approach:

Bertino et al. have proposed a framework based on anomaly detection techniques to detect malicious behavior of database application programs. The approach is as follows. At first step, a fingerprint of an application program based on SQL queries is created. Then, take advantages of association rule mining techniques to extract useful rules from these fingerprints. These rules depict normal behavior of the database application. Finally, dynamic queries check against these rules to detect injection attacks that not conform to these rules.

2) STUDY ON SQL INJECTION ATTACKS: MODE, DETECTION AND PREVENTION, Vol. 1, Issue 8, ISSN No. 2455-2143, Pages 23-29, International Journal of Engineering Applied Sciences and Technology, 2016

The following dynamic approach was used in this research work.

Its two stages are:

a) Frontend Phase

b) Backend Phase

Frontend Phase

Initially at front end we secure Database from any SQLIA. In this methodology we include an additional section in client table to store the Final Hash Code, which is obtained during enrollment time of a client for the first time and is put into client table along with client name and secret key as demonstrated in the Table - 1.

CLIENT	SECRET KEY	FINAL HASH CODE
Ritu	Ritu09	12HJHOP34TTM
Geeta	Geeta123	21313NKIFIFN3

Table - 1: Client Table

When a client logins,, a genuine client is recognized by matching of client name, secret key and final hash code which was generated at run time using the stored procedure. For final hash code computation we will continue as indicated by the given architecture in the Figure – 1.

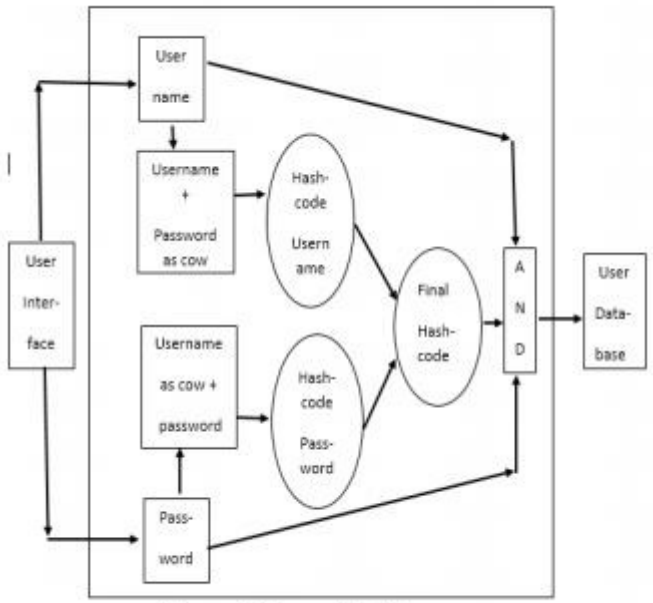


Figure 1: Proposed Architecture

Working Methodology:

The working of proposed methodology is defined in two kinds:

1) New Client Registration: A new client enters the log in details like distinctive name and secret key on client side to get registered. As indicated by the proposed design, the distinctive name and secret key is prepared at the center level. Below are the stages:

1. To discover hash value of log in name by secret key as cow.
2. To discover hash value of secret key by log in name as cow.
3. Linking the result of step1 and step2 to discover final hash code.
4. Login name, Secret key and final hash code are to be put away into the client table.

2) Login and verification: The login structure must be filled by the client to get signed into the database. Below are the given stages:-

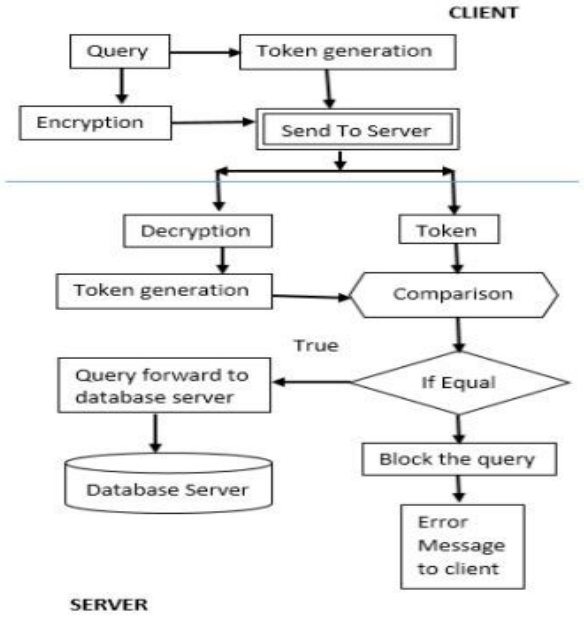
1. A distinctive name and secret key is to be entered at client side.
2. The name put away in client table is matched with the entered client name.
3. As per proposed system to discover final hash code at run time, the client name and secret key is handled after the client name is being matched.
4. Final hash code and secret word is checked with stored values in the database.
5. In the event that client is legitimate then he/she can get to data from database or else incorrect text is shown.

Backend Phase:

The proposed framework notices on how SQLIA on Web applications by tokenization and encryption for detection and prevention. The tokenization process changes over the input query in fruitful token and dynamic table stores it at the user end. Name of field, name of table and information are encoded by AES algorithm is connected by recognizing spaces on the data query, double dashes and single quotes, and so on.

The initial encrypted query and table which is tokenized is being sent on the server side. Now the query is decrypted and generated into number of tokens which are then stored into other dynamic table at the server end. After comparing both the dynamic tables, if they are same then it is evaluated that there was no injected query, henceforth the query is carried to the central database for fetching the output. In the event that they are distinctive, query is dismissed and not sent to the server of the database. The incorrect text warning is sent to the client.

1. **AES Encryption or Decryption:** The data and attributes of the query are encoded by AES (Advanced Encryption Standard) algorithm needs less storage and this process is quick [14]. As soon as the query is received at server end, it gets decrypted with the similar key and gets transformed into different tokens which are kept in the other dynamic table.



2. T**okenizing the query:**

   In this strategy, the tokens are being created from query input given by the client. All strings before a space, a single quote, and double dashes constitute a token. It is carried out in four stages:-

   Step 1: All the unimportant characters are exchanged which could have attacked on the query.

Step 2: Identify the query with single quotes, spaces and, doubles dashes. The Figure - 4 demonstrates how tokens are made in data query by recognizing single quote, spaces and double dash for the underneath query: "SELECT e_id, e_name FROM Employee WHERE pay > 1000"

Step 3: The query is broken into different fruitful tokens.

Step 4: The Dynamic table keeps the tokens.

Step 5: Sending query after tokenization, the dynamic token table and the encrypted query are sent to server end.

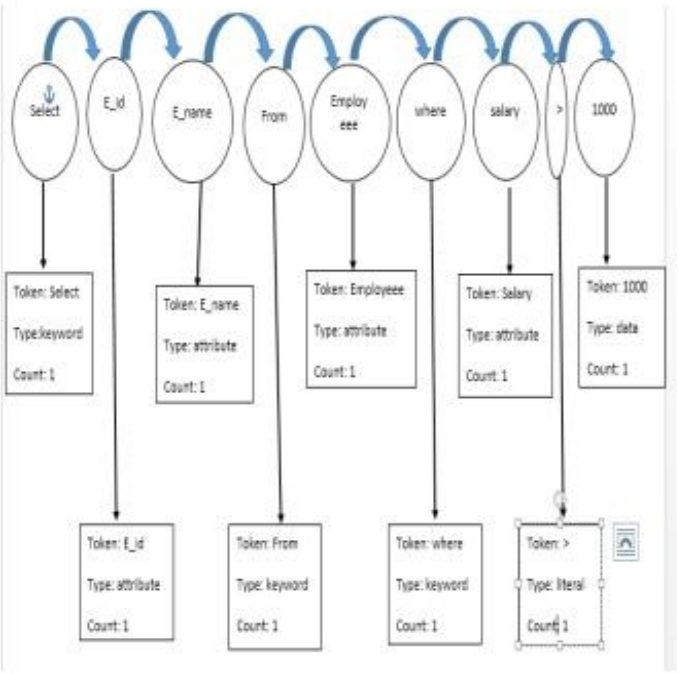


Figure 4: Formation of Tokens

References:

- **http://www.unixwiz.net/techtips/sql-injection.html**
- **classes.soe.ucsc.edu/.../SQL%20Injection%20Attacks.ppt**
- **homes.cs.washington.edu/~suciu/current-trends.ppt**
- **www.cse.iitb.ac.in/dbms/Data/.../DBSecurity-Overview.ppt**