

## 1. Introduction

The main goal to use SQL injection attack includes illegal access to a database, extracting information from the database, modifying the existing database, escalation of privileges of the user or to a malfunction of an application. In the end, SQLI includes unauthorized access to a database exploiting the vulnerable parameters of a web application.

So, in this chapter we take up the existing mechanisms to preventing SQL Injection. Solutions addressed to prevent SQL Injection Attack include existing defensive coding practices of encrypted algorithms. Hence, in this chapter, we talk about existing solutions that only give more information which help us identify some mistakes in order to avoid them in our application.

## 2. Existing solutions to prevent SQL injection using a cryptography system

### 2.1. Mixture Secure Hashing technique and Advanced Encryption Standard (AES)[2]

Neha Mishra and Sunita Gond proposed a technique to prevent SQLI (SQL injection) by using a cryptography system in addition to a strategy which relies on a mixture of secure hashing technique and AES to secure the database against SQLI attacks.

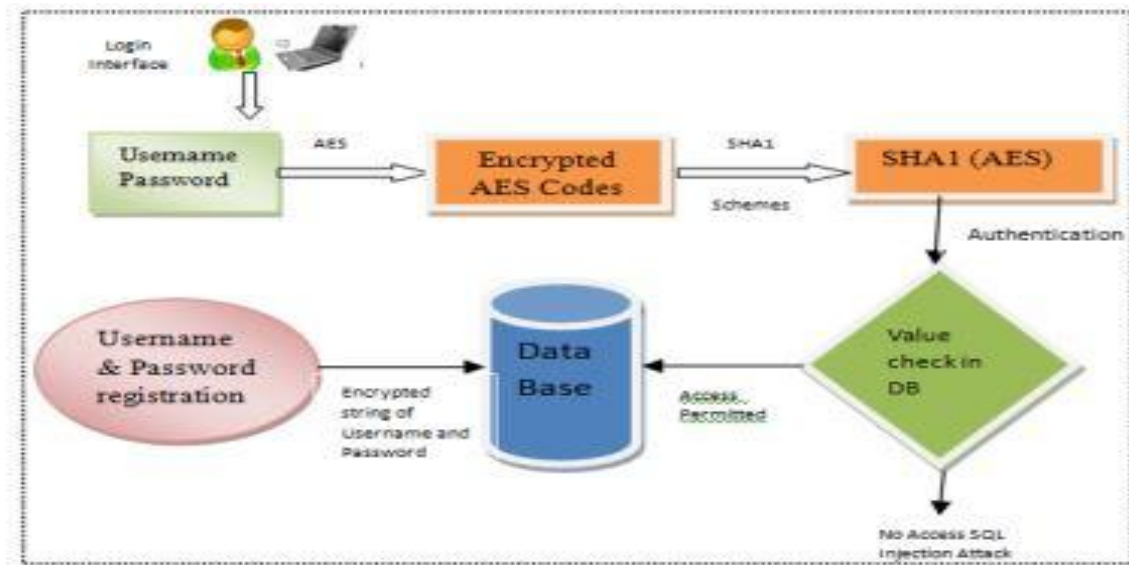
As we show in the table below, in the login table there are two columns: one for username and another for password. In this strategy, we need more columns: one for Secure hash value of username and one for secure hash value of password.

The secure hash values of username and password are stored in the login table when the user creates an account for the first time.

#### 2.1.1. Login phase

- ✓ When the user wants to login, his identity is verified via the username and password. The hash values are also verified.
- ✓ In the database there is another table for AES and a combined approach of secure hashing on encryption. The reason behind use AES is to change the data into a form that is not readable by the users (hackers) without the correct key.

The goal of this technique is that this format cannot be deciphered till the hacker attacks the information. So by this way hackers cannot bypass the authorization to access database.



**Fig.3.1.**AES and secure hashing technique[2].

## 2.2. Preventing SQL injection attacks using Blowfish and RSA[26]

Aakash Ahuja, Pulkit Arora, Shashank Singh propose a model for preventing SQL injection attacks by combining two encryption techniques Blowfish and RSA.

During the registration process, the new user needs to provide a username and password. Along with its regular process of checking the availability of username, on successful completion, the server generates the user key, which is hexadecimal value of the password and stores it in the user's table along with username and password as shown in the Table.3.2.

The information of the registered users is stored in the server database, in the user table with other user details like username, password and the key. Once the user is registered he can access the database by requesting a login to the server and the server respond with the access control once the requests is validated, there are two stages in the process:

### 2.2.1. Access Request Process

Access is provided to the database by the server after verifying the user's authentication. Every new user needs to register with the database first. Existing users can directly access the database by providing their username and password. The steps are as follows :

#### 2.2.1.1. Registration

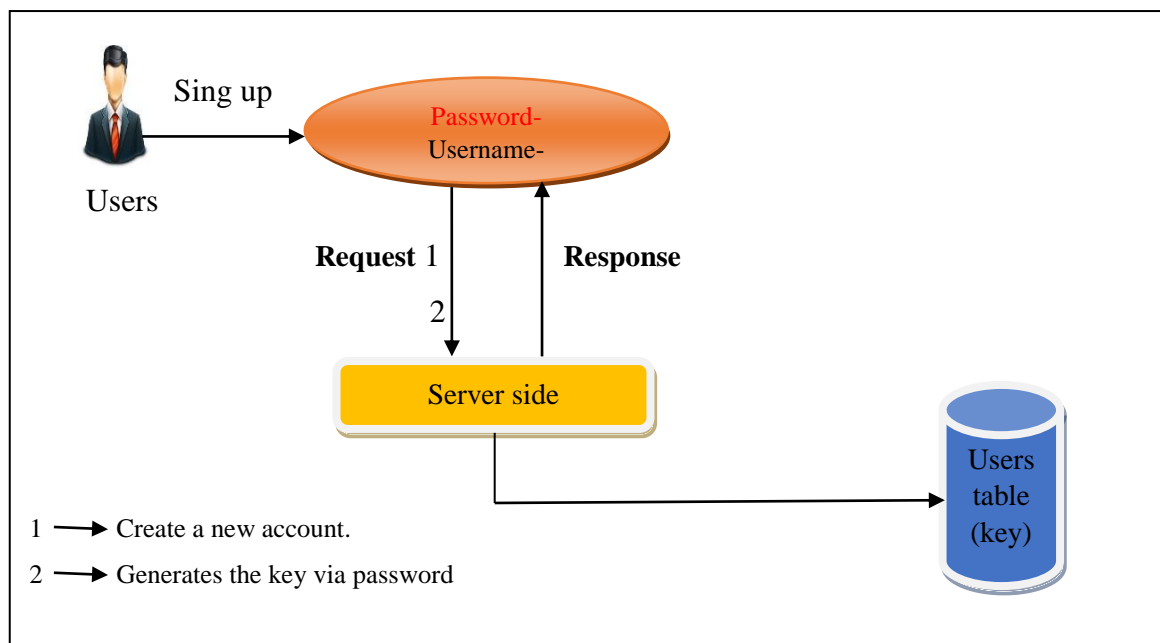
- ✓ A new user has to register himself/herself with the database server.
- ✓ For registration, a user has to provide a valid username and password of a minimum of 4 characters.

- ✓ The users also has to fill in a randomly generated CAPTCHA to ensure that it is a human try or a computer generated try, to prevent hackers creating malicious accounts.
- ✓ The hexadecimal value of the password provided by the user is stored in the user authentication table, which will act as a key for blowfish decryption.

#### 2.2.1.2. Login

- ✓ User enters username and password.
- ✓ Password is converted into hexadecimal.
- ✓ Username and password are encrypted using the hexadecimal value of the password as a key, by the blowfish encryption technique.
- ✓ A SQL query is generated for the user request with username, password and encrypted username and password.
- ✓ SQL query is then encrypted with RSA encryption using a public key for further security.
- ✓ Encrypted query is then sent over to the server. Since a two level encryption technique is used in this scheme the level of security is very high. The server end has to decrypt and verify the user's authentication and provide further access to the database.

Here also we can put simple client validation such as simple application which removes special characters and limits the size of input, as we mentioned above in chapter 1.

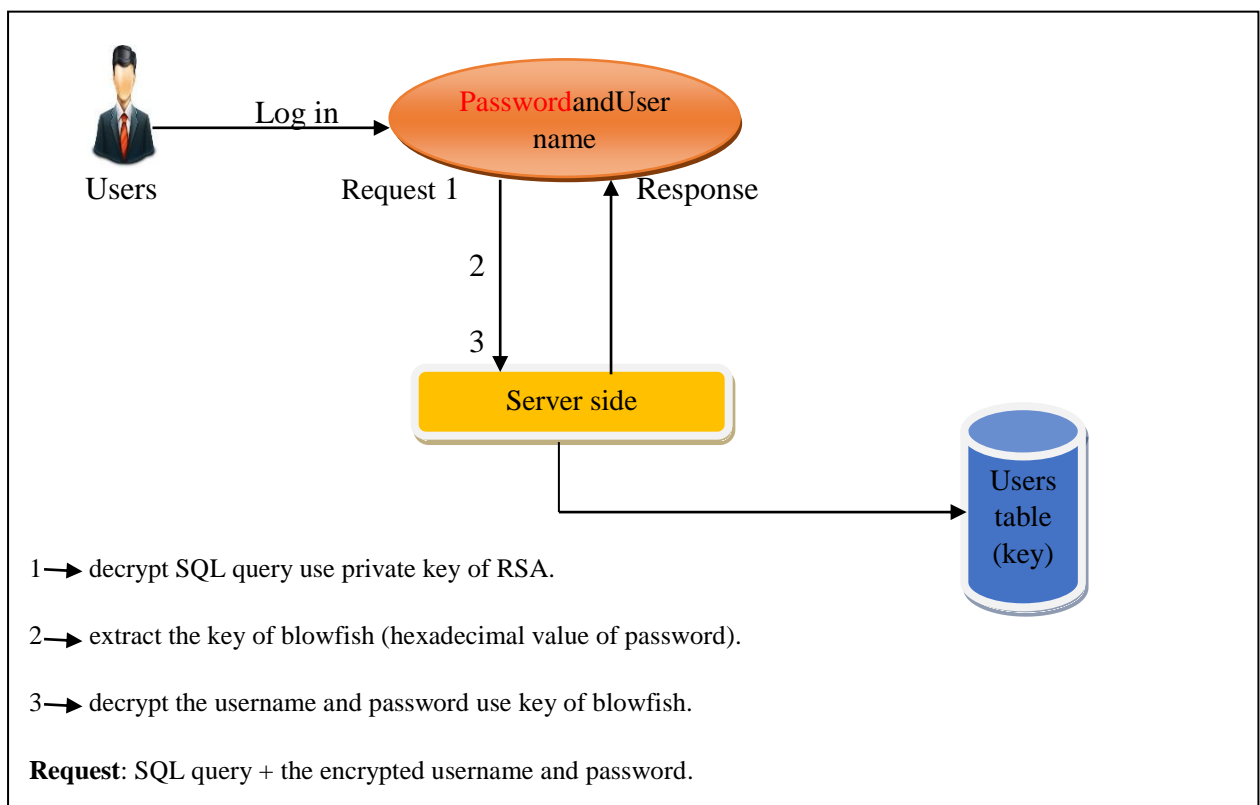


**Fig.3.2.**Registration phase.

### 2.2.2. Access grant process

The server provides access to the requested user only after the verification is complete. The following steps are included:

- ✓ The server receives encrypted data.
- ✓ The data is decrypted to get SQL query using a private server key.
- ✓ From the decrypted SQL query the username and password are used to bring the key (hexadecimal value of password) stored in the user authentication table.
- ✓ A key is used to decrypt the encrypted username and password in the query.
- ✓ If the decrypted username and password match any of the authorized users in the user table stored in the server then access is granted to the user or else the query is rejected. Blowfish algorithm will only be able to decrypt successfully the data if the hexadecimal value of the password and the hexadecimal value stored in the user's table match.



**Fig.3.3.**login and Access grant process.

## **2.3. Preventing SQL injection attacks using Multi-hashing[27]**

### **2.3.1. Definition**

In this research, Yogesh Bansal and Jin H. Park. Suppose an authentication process of a web application that relies on mutlti level hashing, and a couple of databases, product and query databases:

- ✓ Product databases: include the first hash code(username and password).
- ✓ Query databases: include the second hash code (predetermined login query with the first hash code).

The main goal of this strategy is to use the hash function to remove an HTML tag from the user's input and the user's regular expression search. This proposed strategy consists of three steps: the registration phase, login phase and validation phase.

### **2.3.2. Registration phase**

The new users<sup>2</sup> needs a combination of username and password to sing up in a user interface. Any information that comes from the user is directly sent to the server side that inputs validation operations.

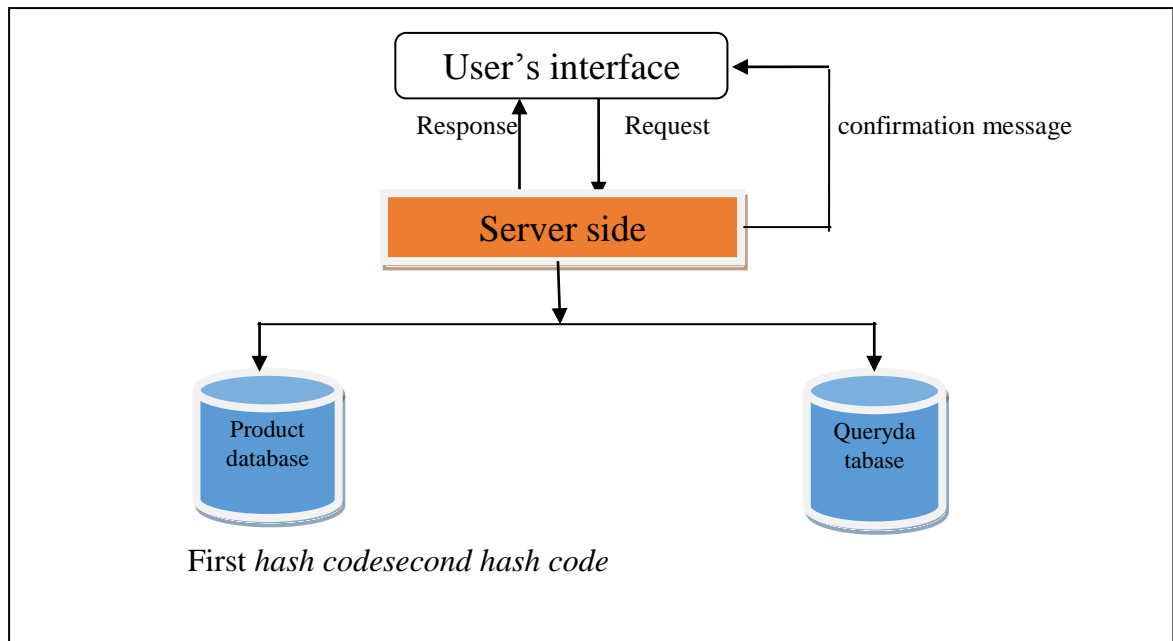
#### **2.3.2.1. First hashcode**

By using the SHA-512 encryption algorithm, the server side generate the first hash code through a couple of username and password. where they stored the hash value in product databasewith other information as: email and phone number.

#### **2.3.2.1. Second hash code**

We implement the same hash function to predetermined login query with the first hash code. The second hash code and the predetermined login query information are stored in the query database, which is separated from the product database for modular design.

The last operation in the registration phase is that the server sends back a confirmation message to the client (user interface).



**Fig.3.4.**Registration phase.

### 2.3.3. Login phase

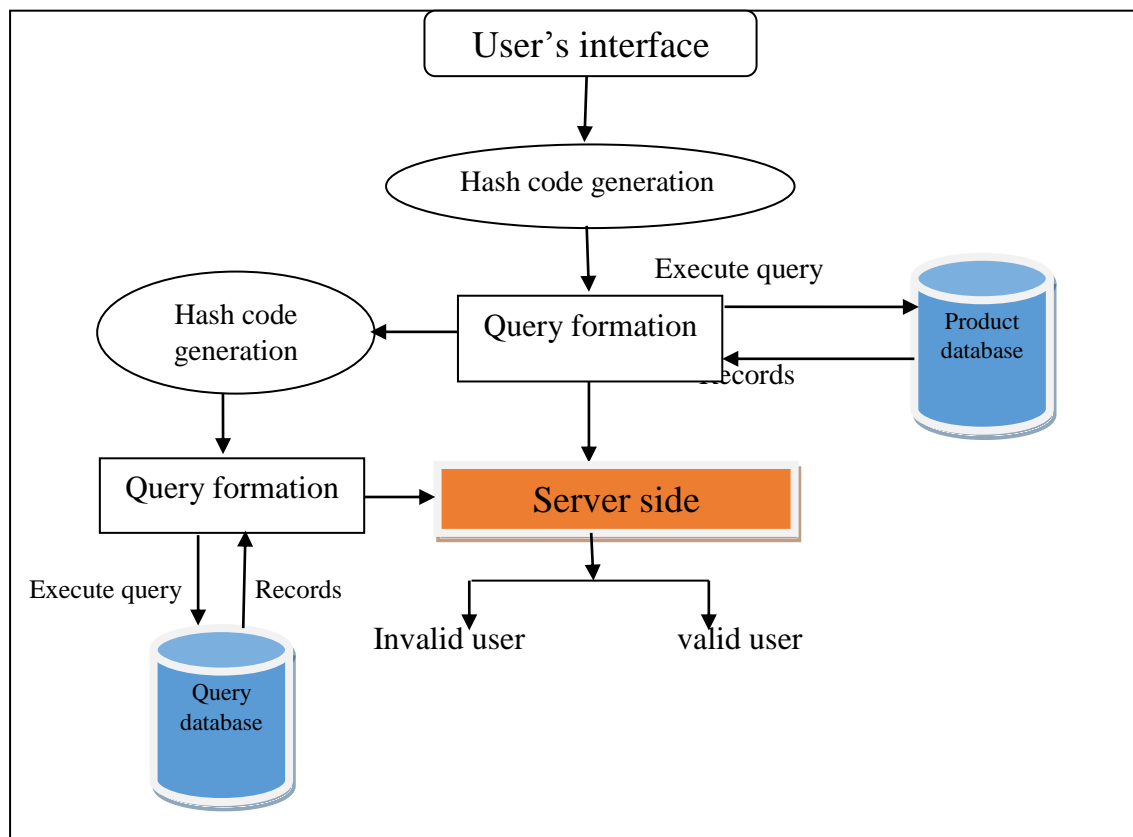
When a user logs in, the username and password are passed to the server, and the server generates a hash code, first hash code, by applying the SHA-512 encryption algorithm as it is done in the registration phase. This hash code is used to get the user's information (records) from the product database, and is used with the login SQL query to generate the second hash code via the same encryption algorithm.

The security gain from using the first hash code to access the product database, instead of using the username and password directly, is to avoid to deal with naturel data. The second hash code will be used in the validation phase to apply the security of the web application against SQLI (SQL injection) attacks.

### 2.3.4. Validation phase

The validation phase is related to the login phase as the two hash codes, which are generated in the login phase, are used. When the first hash code is available in the login phase, the user's records are extracted from the product database via a predetermined login query. Since the first hash code is a unique value, it always retrieves a single user's data (records). The retrieved authentication information, username and password, are compared to the login username and password for identity. In the steps bellow we explain the validation process that occurs:

- ✓ Use first hash code as a key to extract the expected result that is stored in the product database.
- ✓ Use second hash code as a key to extract the expected result that is stored in the query database.
- ✓ Compare the number of records which are extracted from the product database with the number of records which are extract from the query database.



**Fig.3.5.** Login and validation phases.

If they are identical, that mean log in successful, otherwise the validation is determined as unauthorized access. For example, if an attacker uses a SQL injection in the login phase to extract information illegally, the number of records from the product database and the number of records stored in the query database do not match and the information is protected and SQL injection attack will be prevented.

#### **2.4. Random4 algorithm[10]**

Srinivas Avireddy, Varalakshmi Perumal proposed random4 algorithm which is based on randomization and is used to convert the input into a cipher text. Any input in web forms will contain numbers, uppercase, lowercase or special characters.

Let us keep in mind the input from the user is encrypted based on randomization. In this algorithm, the valid inputs are numbers, lowercase or uppercase characters and at most **10** special characters. The reason for choosing only **10** special characters is that they are seldom used and for additional security. Each character in the input can have **72** combinations (**26** lowercase, **26** uppercase, **10** special characters). Hence, for a **6** character input there can be **72<sup>6</sup>** combinations possible. To encrypt the input, each input character is given four random values.

Example:

	R [1]	R [2]	R [3]	R [4]
a	;	l	x	W
z	7	k	@	U
A	i	J	)	0
Z	M	6	f	.
0	9	B	g	“
9	c	R	j	l

**Table.3.3.**Table lookup table for random4.

A sample lookup table is given in **Table.3.3**. Based on the next input character, one of these four values is replaced for a given character. For example, let username be the input to be provided. If the username is “abc”, the first character ‘a’ is given **4** random values in the lookup table. Based on the next input character which is lowercase ‘b’ here, the value R[1] is chosen. If the next character was uppercase R[2] would have been chosen and R [3] if it was a number and R[4] if it was a special character or no character.

### 3. Conclusion

In this chapter, we dealt with the existing mechanisms which use cryptography systems to prevent SQL injection as (Preventing SQL injection attacks using Multi-hashing, Random4 algorithm...), we also discovered the Strengths and weaknesses for every strategy to find the best algorithm of cryptography. We tried in this chapter to find another technique to prevent SQL injection which is based in a cryptography system. This idea is applied in the last chapter.