Prevention of SQL injection attacks through encoding technique

Haimashreelakshmi (212IS010)

Department of Computer Science and Engineering
National Institute of Technology Karnataka
Surathkal, Mangaluru, India

Abstract. SQL injection is one of the most serious attack on database related applications.Many researchers have proposed various methods to detect and prevent this attack but are not successfull in stopping attackers completely.By using this attack, a attacker can gain access to sensitive informations like passwords,personal informations etc. Beacuse of this attack organizations face lot of damages in their business.In this project, various types of SQL injection attacks are discussed and its prevention using encoding method is discussed.

Keywords: SQL injection, Attacker, Prevntion, Encoding

1 Introduction

SQL injection is an attack that allows attacker to manipulate query that an application inputs to its database. This allows the attacker to access, read and modify sensitive data.This might be data that belong to other users, only the administrator or only to organization users etc. If the attacker modifies or deleted these unauthorized data it results in permanent changes to applications behaviour.There are different types of SQL injection attacks:

1. **In-band SQLi:** Here same communication channel is used by attacker to perform attack and also to get results.
   - **Error-based SQLi:**Here when database produces errre messages as a result of attackers action information about the database is collected by the attacker from these error messages.
   - **Union-based SQLi:**Here attacker uses UNION operator of SQL to get desired data with additional queries in single HTTP response.Resulting in modification to database.
2. **Blind/inferential SQLi:**
   - **Content/boolean-based SQLi:**Here attacker forces application to return result depending on malicious query result as true or false.Attacker understands if result in HTTP response remains same or changes.
   - **Time-based SQLi:**Here attackers query foreces applications to send result after specific time so that attacker can decide if its true or false depending on the response time.
3. **Out-of-Band SQLi:**This is used by attacker when he cannot perform attack and gather its result on same channel.

1.1 Problem Description

SQL injection results in serious security related issues.In this project study on various SQL injection attacks is done and prevention using encoding technique is implemented.

1.2 Motivation

SQL injection is one of the serious threat in database applications.To protect database from this attack prevention methods must be developed. Here SQL injection attack is prevented using encoding technique.

1.3 Scope

The method proposed can be used to prevent SQL injection attack in database applications.

1.4 Objectives

1. Study on various types of SQL injection attacks.
2. Understanding how SQL injection attack is done by attacker.
3. Preventing SQL injection attack using encoding technique.

1.5 Organization of the Report

The next part of this report consists of literature survey as section 2, proposed approach as section 3, experimental results as section 4 and finally a conclusion.

2 Related Work

In this paper a technique is implemented where developers query result size is dynamically analyzes and then attack is detected by comparing it with actual query result [1]. The detection of SQL injection is done using graph tokens and SVM classifier [2]. Here tokens are generated from query then based on their interaction graph is constructed. SVM classifier is used to the queries during runtime. Aliero et al. proposed protection method against SQL injection where instruction set randomization is used to create various data instances which are not known to the hacker [3].
Halfond and Orso proposed methodology to identify and stop injurious queries in advance and developed tool called AMNESIA [4]. Bandhakavi et al. proposed method to prevent SQL injection attacks [5]. He used dynamic candidate evaluations where automatically web applications were transformed to be safe against various SQL injection attacks. Halfond did a review on various different types of SQL injection methods and how the attack takes places and its strength and weakness [6].

Alwan and Younis did a survey on various SQL injection attacks and presented classical and modern types of SQLIA and various existing techniques to detect and prevent these attacks [7]. Alsahafi did a study on various detection and prevention technique of SQL injection attacks and compared them and analyzed their performance [8]. Pietraszek and Berghe developed a CSSE(context sensitive string evaluation) tool for PHP platform. This platform is prone to vulnerabilities and phpBB was used to validate method. CSSE detected and prevented all the SQL injection attacks [9].
Paper by Lee et al. proposed a very simple and effective method to identify SQL injection attacks in web applications by comparing SQL queries with dynamically generated no attribute queries [10]. This method is also used in databases that are connected to applications. Wei et al. proposed a method prevent penetration attack in stored procedures [11]. Halfond and Orso developed a technique that uses static analysis and runtime monitoring combines to detect and stop illegal queries before they are executed on the database. In its static part, the technique builds a conservative model of the legitimate queries that could be generated by the application [12]. In its dynamic part, the technique inspects the dynamically generated queries for compliance with the statically-built model [12].

3 The Proposed Approach

In the proposed approach a website is built using HTML,PHP and MYSQL database.Apache web server is used for hosting.In the website there is a option to choose admin or user.If admin is choosen then admin login page is displayed where admin needs to enter userid and password.If its authenticated correctly then admin can create new users and add it to database.If user is choosen then user login page is displayed where user needs to enter valid userid and password given to them.If these credentials are found to be correct then user can view his personal details.
In the database, two tables are created namely admin and user_details.In admin table userid and password of admin is present that is usd to authenticate the admin.In user_ details table various details related to user like first name,last name,gender,country and email address are present along with userid and password.In both of these tables userid and password together forms primary key. In this system, if userid and password are stored in plaintext format then if any malicious entries are done at user end then unauthorized user can access the sensitive information present in the database.For example in the above system, if attacker wants to get all the information about users present in user_details table then he can enter some userid for example 'abcd' and then in password he enters anything' or 'x'='x.
The actual query,
SELECT * FROM user_details WHERE userid='$uid' AND password='$pid';
gets converted to malicious SQL query,
SELECT * FROM user_details WHERE userid='abcd' AND password=' anything' or 'x'='x';

Now all the user details will be displayed to attacker because based on operator precedence, the WHERE clause is true for every row, therefore the query will return all records.The example given here is an tautology based attack.In a tautology-based attack, the code is injected using the conditional OR operator such that the query always evaluates to TRUE [13]. Tautology-based SQL injection attacks are usually bypass user authentication and extract data by inserting a tautology in the WHERE clause of a SQL query [13]. The query transform the original condition into a tautology, causes all the rows in the database table are open to an unauthorized user [13].
Same thing happens if this is done in admin login as well.Attacker can authorize himself as admin and enter any details as he wishes to user_details table.
The actual query,
SELECT * FROM admin WHERE userid='$userid' AND password='$pass';
gets converted to malicious SQL query,
SELECT * FROM admin WHERE userid='abcd' AND password=' anything' or 'x'='x';
Now attacker enters to system as admin and can add any details to user_details.
But if userid and password are stored in encrypted format then base64 encryption for malicious entries will be different from actual userid or password because of which SQL injection can be prevented. In case of user login,
SELECT * FROM user_details WHERE userid='YWJjZA==' AND password= 'YW55dGhpbmcnIG9yICd4Jz0neA==';
In case of admin login,
SELECT * FROM admin WHERE userid='YWJjZA==' AND password= 'YW55dGhpbmcnIG9yICd4Jz0neA==';
These entries doesn't match with any of the database entries and hence attacker is unsuccessful in his attempt to enter system.
There are other types like piggy backed queries where the hacker injects additional queries to the original query, as a result the database receives multiple SQL queries [13]. The first query is valid and executed normally, the subsequent queries are the injected queries, which are executed in addition to the firs [13].Other one is union query where attack can be done by inserting a UNION query into a vulnerable parameter which returns a data that is the union of the result of the original first query and the results of the injected query [13].

3.1 Base64 Encoding Method

In this method to encode, given string is first converted into binary bits.For each of the character in a string ASCII value of that character is taken and is converted into binary. Each of these characters binary representation must have 8 binary bits. Then these binary bits grouped to 6-bit groups. Pad with zero bits at the end to form an integral no of 6-bit groups.Then map every 6-bit group into one base64 character.When the number of characters to be encoded does not come with a multiple of six bits, zeros will be used to complete the last six-bit sequence.To indicate padding append = at the end of the string.
To decode, First remove padding character =.Then convert each base64 character

to 6-bit binary representation.Finally divide these bits into 8-bit chunks and to get ASCII values of characters which gives original string.
Base64 character set has 64 characters namely:

- Upper case alphabet characters A-Z.
- Lower case alphabet characters a-z.
- Number characters 0–9.
- Characters + and /.
- The = character is used for padding.

Each of the above characters are mapped to 6-bit binary representation in base64 encoding mechanism.

4 Experimental Results and Analysis

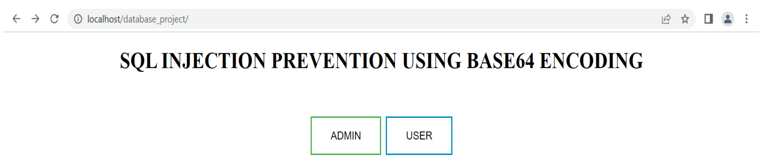
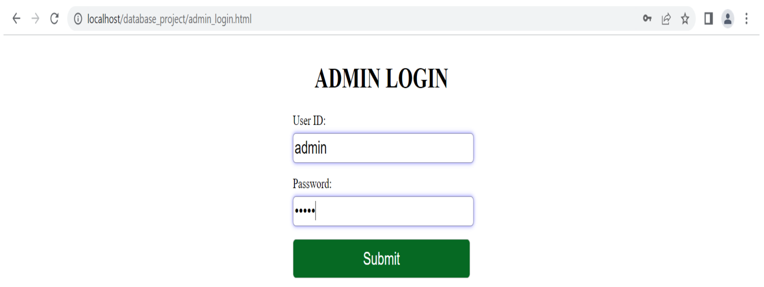

localhost/database_project/
SQL INJECTION PREVENTION USING BASE64 ENCODING
ADMIN
USER

Fig. 1. First page where user can select either admin login or normal user login



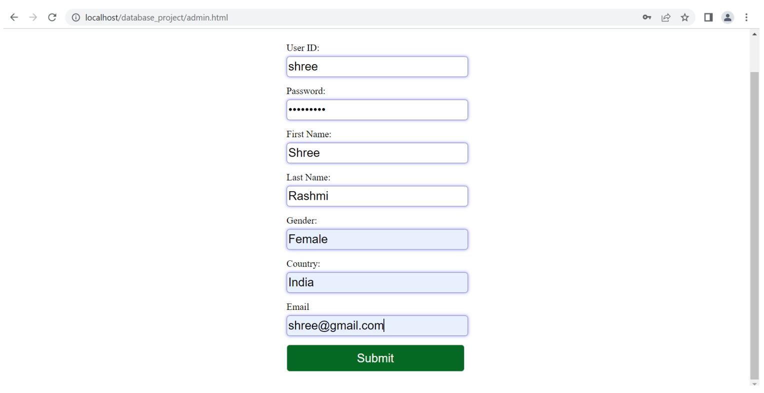
localhost/database_project/admin_login.html

ADMIN LOGIN

User ID:
admin

Password:
•••••

Fig. 2. Admin can enter credentials to access the system



localhost/database_project/admin.html

User ID:
shree

Password:
•••••••••

First Name:
Shree

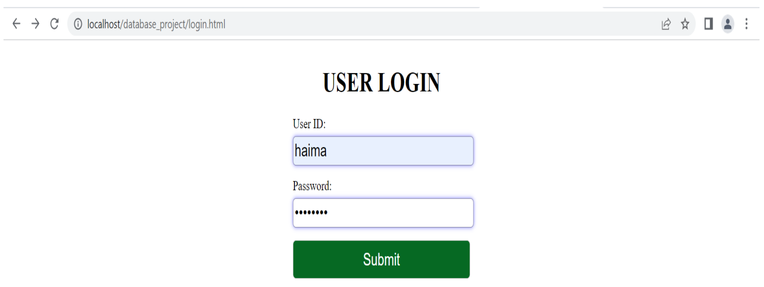
Last Name:
Rashmi

Gender:
Female

Country:
India

Email
shree@gmail.com

Fig. 3. Admin can enter details of new user to database

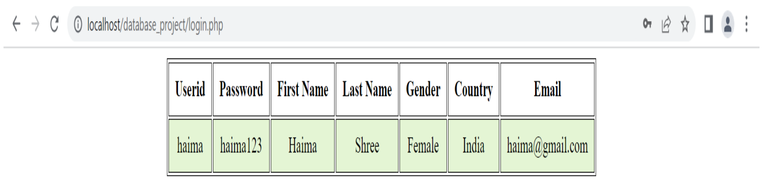

localhost/database_project/login.html

USER LOGIN

User ID: haima

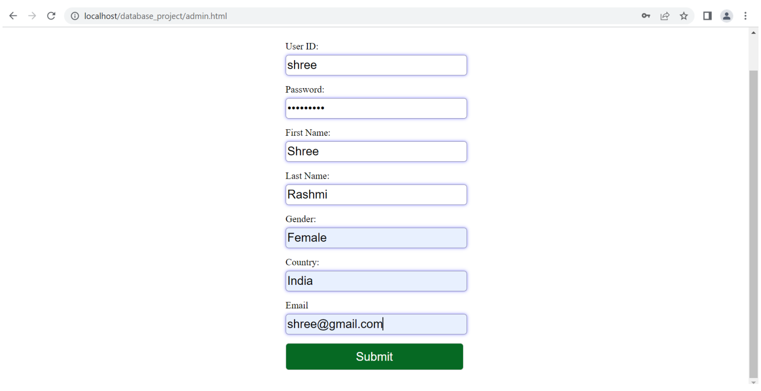
Password: ••••••••

Fig. 4. Normal user can enter his credentials here to get access to his/her details


localhost/database_project/login.php

Userid	Password	First Name	Last Name	Gender	Country	Email
haima	haima123	Haima	Shree	Female	India	haima@gmail.com

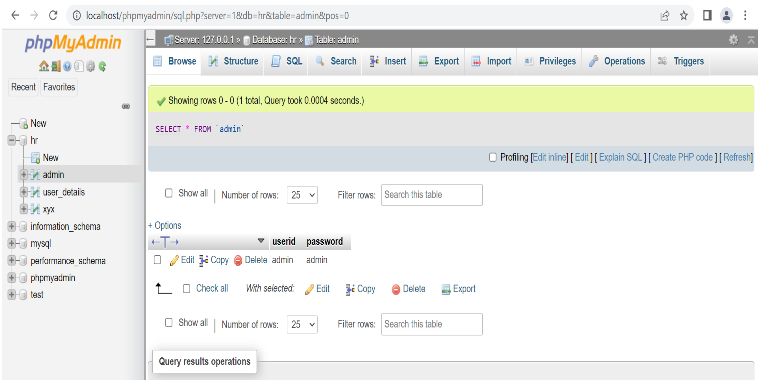
Fig. 5. User details are displayed after checking credentials enterd



localhost/database_project/admin.html

User ID:	shree
Password:	•••••••••
First Name:	Shree
Last Name:	Rashmi
Gender:	Female
Country:	India
Email	shree@gmail.com

Fig. 6. Admin can enter details of new user to database



localhost/phpmyadmin/sql.php?server=1&db=hr&table=admin&pos=0

Server: 127.0.0.1 » Database: hr » Table: admin

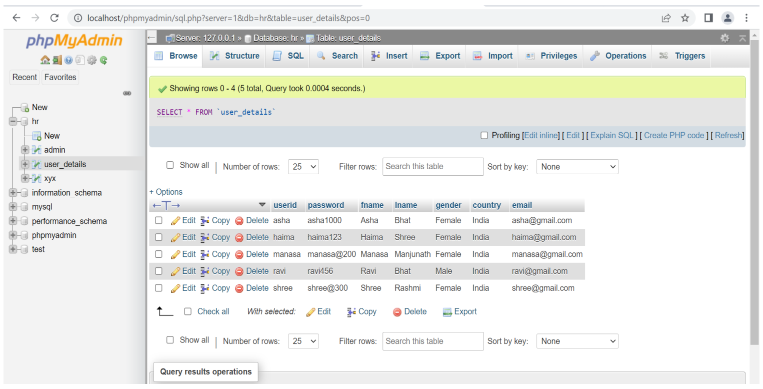
✓ Showing rows 0 - 0 (1 total, Query took 0.0004 seconds.)

```
SELECT * FROM `admin`
```

	userid	password
Edit Copy Delete	admin	admin

phpMyAdmin
Recent Favorites
New
hr
New
admin
user_details
xyx
information_schema
mysql
performance_schema
phpmyadmin
test
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
+ Options
Check all With selected: Edit Copy Delete Export
Show all | Number of rows: 25 Filter rows: Search this table
Query results operations

Fig. 7. Database table admin without encoding



Showing rows 0 - 4 (5 total, Query took 0.0004 seconds.)

```
SELECT * FROM `user_details`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

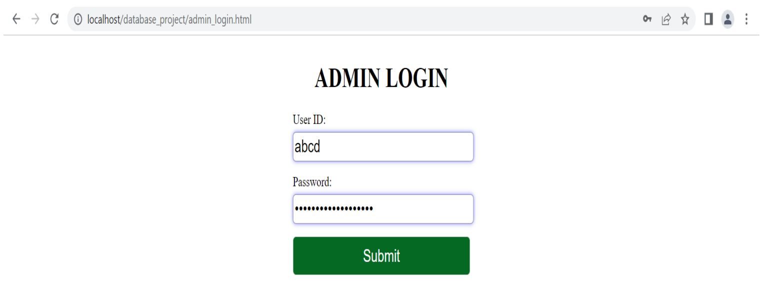
	userid	password	fname	lname	gender	country	email
Edit Copy Delete	asha	asha1000	Asha	Bhat	Female	India	asha@gmail.com
Edit Copy Delete	haima	haima123	Haima	Shree	Female	India	haima@gmail.com
Edit Copy Delete	manasa	manasa@200	Manasa	Manjunath	Female	India	manasa@gmail.com
Edit Copy Delete	ravi	ravi456	Ravi	Bhat	Male	India	ravi@gmail.com
Edit Copy Delete	shree	shree@300	Shree	Rashmi	Female	India	shree@gmail.com

Check all With selected: Edit Copy Delete Export

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Query results operations

Fig. 8. Database table user_details without encoding



ADMIN LOGIN

User ID: abcd

Password: ••••••••••••••••••

Fig. 9. Attacker tries to enter as admin using malicious userid and password

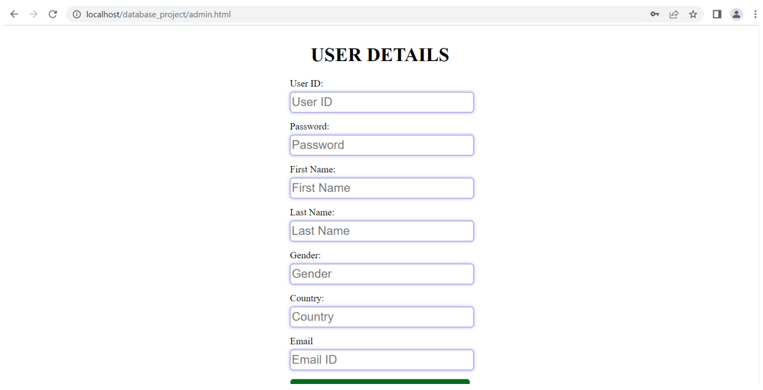


Fig. 10. Attacker logged in as admin and can to entries to database

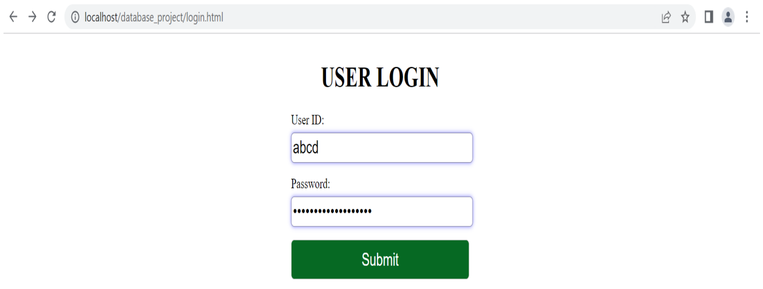
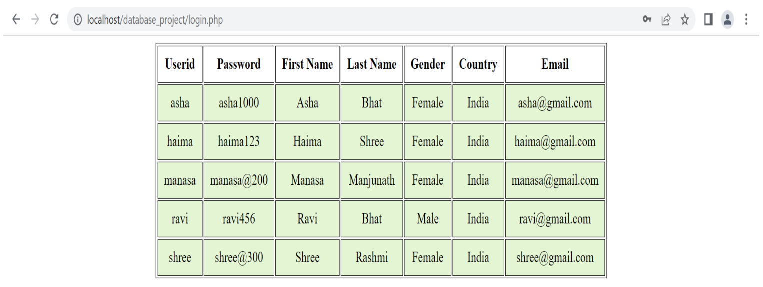
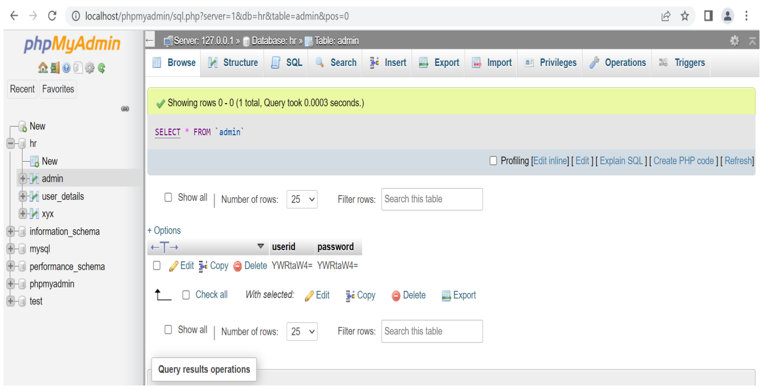


Fig. 11. Attacker tries to get all user details from database using malicious queries



Userid	Password	First Name	Last Name	Gender	Country	Email
asha	asha1000	Asha	Bhat	Female	India	asha@gmail.com
haima	haima123	Haima	Shree	Female	India	haima@gmail.com
manasa	manasa@200	Manasa	Manjunath	Female	India	manasa@gmail.com
ravi	ravi456	Ravi	Bhat	Male	India	ravi@gmail.com
shree	shree@300	Shree	Rashmi	Female	India	shree@gmail.com

Fig. 12. Attacker gets access to all the data in user_details table


localhost/phpmyadmin/sql.php?server=1&db=hr&table=admin&pos=0
phpMyAdmin
Recent Favorites
New
hr
New
admin
user_details
xyx
information_schema
mysql
performance_schema
phpmyadmin
test
Server: 127.0.0.1 » Database: hr » Table: admin
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Triggers

Showing rows 0 - 0 (1 total, Query took 0.0003 seconds.)

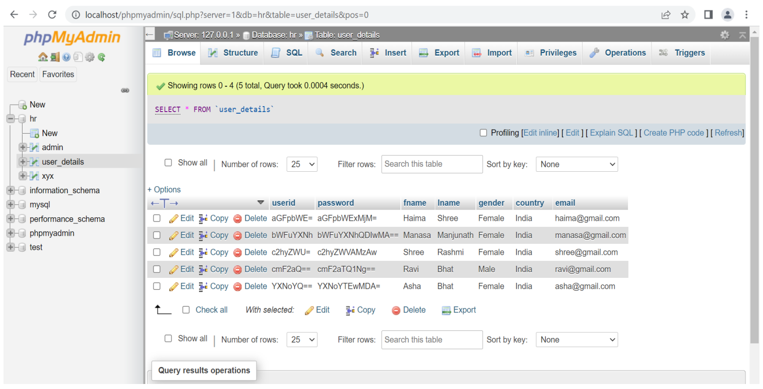
```
SELECT * FROM `admin`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
+ Options

	userid	password
Edit Copy Delete	YWRtaW4=	YWRtaW4=

Check all With selected: Edit Copy Delete Export
Show all | Number of rows: 25 Filter rows: Search this table
Query results operations

Fig. 13. Admin table with base64 encoding of userid and password

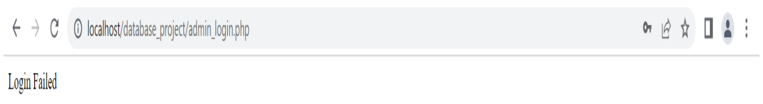


Showing rows 0 - 4 (5 total, Query took 0.0004 seconds.)

```
SELECT * FROM `user_details`
```

	userid	password	fname	lname	gender	country	email
Edit Copy Delete	aGFpbWE=	aGFpbWExMjM=	Haima	Shree	Female	India	haima@gmail.com
Edit Copy Delete	bWFuYXNh	bWFuYXNhQDIwMA==	Manasa	Manjunath	Female	India	manasa@gmail.com
Edit Copy Delete	c2hyZWU=	c2hyZWVAMzAw	Shree	Rashmi	Female	India	shree@gmail.com
Edit Copy Delete	cmF2aQ==	cmF2aTQ1Ng==	Ravi	Bhat	Male	India	ravi@gmail.com
Edit Copy Delete	YXNoYQ==	YXNoYTEwMDA=	Asha	Bhat	Female	India	asha@gmail.com

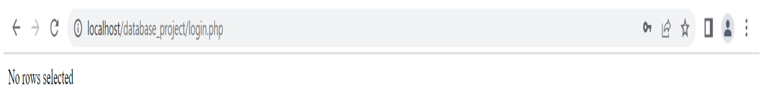
Fig. 14. Table user_details with base64 encoding of userid and password



localhost/database_project/admin_login.php

Login Failed

Fig. 15. Attacker's login is failed as admin when malicious userid or password is given



No rows selected

Fig. 16. Attacker cannot retrieve database data as login is failed as normal user when malicious userid or password is given

5 Conclusion

In this project, various types of SQL injection attacks are studied.Next various methods available to secure database from these type of attackers are examined.In the system built data is secured from attackers using base64 encoding technique.
Some of the future enhancements can be study on blind SQL injection and preventing blind SQL injection. Using some more advanced encoding technique which makes data more safer in the database.

References

1. Y. S. Jang, J. Y. Choi. :Detecting SQL injection attacks using query result size. Computers and Security, vol. 44, pp. 104-118, 2014.
2. D. Kar, S. Panigrahi, S, S. Sundararajan. :SQLiGoT: Detecting SQL injection attacks using graph of tokens and SVM. Computers and Security, vol. 60, pp. 206-225, 2016.
3. M. S. Aliero, I. Ghani, S. Zainudden, M. M. Khan, and M. Bello. :Review on SQL injection protection methods and tools. Jurnal Teknologi, vol. 77, no. 3, 2015.
4. W. G. Halfond, A. Orso. :AMNESIA: analysis and monitoring for NEutralizing SQL-injection attacks. In Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering, ACM, pp. 174-183, 2005.
5. S. Bandhakavi, P. Bisht, P. Madhusudan, V. N. Venkatakrishnan. :CANDID: preventing sql injection attacks using dynamic candidate evaluations. In Proceedings of the 14th ACM conference on Computer and communications security, ACM, pp. 12-24, 2007.
6. W. G. Halfond, J. Viegas, A. Orso. :A classification of SQL-injection attacks and countermeasures. In Proceedings of the IEEE International 13-15, IEEE Symposium on Secure Software Engineering, vol. 1, pp. 1-10, 2006.

7. Zainab S. Alwan, Manal F. Younis. :Detection and Prevention of SQL Injection Attack: A Survey. International Journal of Computer Science and Mobile Computing, Vol. 6, pg.5 – 17, 2017.
8. Raniah Alsahafi. :SQL Injection Attacks: Detection And Prevention Techniques. International journal of scientific technology research, vol. 8, 2019.
9. T. Pietraszek, C. V. Berghe. :Defending against injection attacks through context-sensitive string evaluation. In International Workshop on Recent Advances in Intrusion Detection, pp. 124-145, 2005.
10. I. Lee, S. Jeong, S. Yeo, J. Moon. :A novel method for SQL injection attack detection based on removing SQL query attribute values. Mathematical and Computer Modelling, vol. 55, no. 1, pp. 58-68, 2012.
11. K. Wei, M. Muthuprasanna, S. Kothari. :Preventing SQL injection attacks in stored procedures. In Australian Software Engineering Conference (ASWEC'06), IEEE, pp. 1-8, 2006.
12. W. G. Halfond, A. Orso. :Combining static analysis and runtime monitoring to counter SQL-injection attacks. In ACM SIGSOFT Software Engineering Notes, ACM, vol. 30, no. 4, pp. 1-7, 2005.
13. https://www.w3resource.com/sql/sql-injection/sql-injection.php
14. https://medium.com/swlh/powering-the-internet-with-base64-d823ec5df747