

# Securing Credentials from SQL Injection Attack Using Encryption and Hashing

MSc Internship  
Cyber Security

Farhaan Kaleem  
x17169003

School of Computing  
National College of Ireland

Supervisor: Mr Imran Khan

**National College of Ireland**  
**MSc Project Submission Sheet**  
**School of Computing**



**Student Name:** Mr. Farhaan Kaleem  
**Student ID:** x17169003  
**Programme:** MSc. In Cyber Security **Year:** 2018-2019  
**Module:** Academic Internship  
**Supervisor:** Mr. Imran Khan  
**Submission Due Date:** 12/08/2019  
**Project Title:** Securing Credentials from SQL Injection Attack Using Encryption and Hashing  
**Word Count:** 6,036 Words **Page Count:** 17 Pages

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

**Signature:** .....

**Date:** 10<sup>th</sup> August 2019

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
<b>Attach a Moodle submission receipt of the online project submission,</b> to each project (including multiple copies).	<input type="checkbox"/>
<b>You must ensure that you retain a HARD COPY of the project,</b> both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# Securing Credentials from SQL Injection Attack Using Encryption and Hashing

Farhaan Kaleem  
x17169003

## Abstract

In our daily life, we use web applications for most of the purposes like banking, food ordering and online shopping. Most of the web applications hold the information like our username and password, using which the card details and other such sensitive information can be acquired. According to OWASP, today most of the web applications are vulnerable to SQL Injection Attack. SQL Injection attack can show the sensitive data like passwords on the screen that can be used by other unauthorised users. These passwords are usually hashed, but from the hash we can identify the algorithm used and hence break it. This paper aims the full stack web application developers, who use SQL database for storing the credentials. This paper proposes a system in which the user credentials are not only hashed, but also encrypted. Also the system filters the input, so that no malicious code can be inserted into the database to prevent SQL Injection Attack. The system uses the Argon2 hashing algorithm that was the winner of Password Hashing competition held in July 2015.

## 1 Introduction

We use web applications for most of the things today, wherein we give our personal information to the websites. Hence it becomes necessary for the developer to protect the data from the hackers. OWASP (Open Web Application Security Project) 2017 (“Top 10-2017 Top 10 - OWASP,” n.d.) ranks SQL Injection as the number one vulnerability in the Websites. This means that most of the websites available today are vulnerable to SQL Injection. SQL Injection is possible, if the websites build the SQL queries from the user provided input. Hence, in order to get all the information stored in the database, the hacker manipulates the user input to build malicious queries. (Sajjadi and Tajalli Pour, 2013) describes the different threats that can arise due to the alteration of SQL queries.

Hackers use SQL Injection to attack the database and gain all the sensitive information. This sensitive information can contain the user’s password, credit card details and more such information that if fallen into wrong hands can cause huge impacts. Recently, there was an SQL Injection attack on Capital One Financial Corp. that impacted 106M people (“Capital One Data Theft Impacts 106M People — Krebs on Security,” n.d.). The passwords from the databases are used to study the pattern that the people use to make the passwords. Then these patterns are used to make the dictionary and the tools like John the Ripper uses the same

technique to break the passwords (Jake, 2017). In order to protect the passwords, we generally store the passwords in hashed format.

The hashed passwords can be reverse hashed using the Rainbow Table attack. Rainbow table is the pre-computed table that is used to reverse the hashes of the passwords (“Rainbow table,” 2019). (Kumar et al., 2013) describes that how the Rainbow Table can be used to crack the passwords. It also states that using the rainbow table, the password can be cracked in less amount of time as compared to the Brute Force attack. Hence, we can see that it is possible to crack the password from the hash. Hash itself says us that which algorithm is used to hash the password.

<b>Prefix</b>	\$1\$	\$sha-1\$	\$sha-5\$	\$sha-6\$	\$2\$	\$scrypt\$	\$argon2\$
<b>Hashing Algorithm</b>	MD5	SHA-1	SHA-256	SHA-512	Bcrypt	Scrypt	Argon2

**Table 1: Prefix of Hashing Algorithms**

The hashing algorithm used in this paper is Argon2 hashing algorithm. Argon2 is the newest hashing algorithm and the winner of the Password Hashing Competition 2015 (Hodgkiss, 2018). It is very memory intense and hence avoids password cracking by GPU and ASIC which are the fastest password cracking techniques used today (Hodgkiss, 2018). Argon2 has two versions: Argon2d and Argon2i. Argon2i is recommended for hashing.

The focus of this paper is to secure the hashed password by using AES encryption algorithm. Also the paper proposes a website that uses SQL database at the backend. The website imposes restrictions on the users like length of the password, including letters, numbers and special characters in the password. Also it does input filtering to avoid SQL Injection attack. The password given during registration is hashed using Argon2 algorithm. This hash is then encrypted using AES algorithm and stored in the database. This ensures that even if SQL Injection attack takes place, the password hash is highly secured so that no sensitive information is revealed.

## 2 Related Work

Subsection 2.1 describes the different vulnerabilities present in SQL, leading to SQL Injection attack. Subsection 2.2 gives us the knowledge about the various techniques and methods that can be used to do the analysis of the password patterns. Password patterns can be obtained from the hashes of the passwords. From the password patterns, it is easy to crack the password. Cracking of password needs a large amount of time, but with the advancement in technology and the different techniques used, this time is reduced to a large extent. These techniques are discussed in subsection 2.3. Subsection 2.4 shows us the different techniques used for securing the user credentials.

## 2.1 SQL Injection

SQL Injection Attack or SQLIA ranks number 1 according to the OWASP top 10 (“Top 10-2017 Top 10 - OWASP,” n.d.), which means that most of the websites available today are vulnerable to SQLIA. SQLIA can be used to display the whole database on the screen, which may include the sensitive data of the users including bank details. According to (Kindy and Pathan, 2011), if one is able to perform SQL Injection on the web application, it is equivalent to have access to whole database of the application. Hence, we can see that it is very important to protect the web application from SQLIA; else all the data will be open to the hackers. To prevent SQLIA, it becomes necessary to know that how SQLIA takes place. Below table shows the different types of SQLI attacks described by (Kindy and Pathan, 2011).

SLIA Technique	Explanation
Tautologies	Queries, which executed always gives output as true.
Error Messages	The errors thrown by the database are studied to understand the database used and its underlying query structure to break it.
Nested Queries	One query inside another query to perform multiple actions with a single query.
Union Concatenation	Keyword ‘UNION’ can be used to combine the queries and get the data from the multiple tables in the database.

**Table 2: Types of SQL Injection Attack**

(Johari and Sharma, 2012) describes the various prevention techniques that can be used to prevent the SQLIA. However, it also says that almost all of the techniques have their own drawbacks such as:

- Frameworks used are very complex
- Runtime overhead on the processors
- Irregular or incomplete implementation
- Need of huge amount of manual work

Also it says that the first tier of defence is web application itself and hence it is the responsibility of the developers to use the good coding techniques.

## 2.2 Study of Password Patterns

(Tatlı, 2015) describes the various issues that the database and the passwords stored in them are currently facing. The paper also states the common mistakes that the developers do while coding. These mistakes includes saving the password in plain text format or hashed format. Once the hashes are in the hands of the hackers, they use different techniques to obtain the original password from the hash. These techniques include Brute force attack, dictionary

attack or rainbow table attack. The author himself has worked on cracking the passwords from their hash values using the Brute force attack. He was able to successfully reduce the amount of time required to break the password by using the commonly used passwords. Hence from the paper we can see that pattern based brute force attack can easily crack the passwords.

RockYou database was used for the analysis of the password. The analysis was done both using tools and manually to identify the pattern, which was then fed to the password generator. The password generator generated many passwords depending on the pattern received. This password dictionary was used to attack on fifteen different datasets and it successfully cracked the hashed passwords of the database. Hence we can see that the passwords which were assumed to be secure can be cracked easily today. Hence keeping the hashes of the passwords in the database is no longer safe.

### **2.3 Optimized Cracking of Hashed Passwords**

The parallel processors have given the great speed to the computers. (Hongwei Wu et al., 2011) experiments the hash reversal of MD5 algorithm using a multi parallel architecture known as CUDA. CUDA basically includes three components: driver, the runtime library and the library that is used for the calculations. Using CUDA architecture, it is possible to execute large amounts of threads together. The code written for the CUDA consists of two basic modules. One module was used to traverse the input and the other was used to calculate the corresponding MD5 hash. MD5 can be cracked in 64 steps but using CUDA, it was possible only in 49 steps. The same experiment when done using CPU is 16 times slower. The paper concludes that the similar method can be used on the other famous hashing algorithm like the SHA family.

The similar experiment of reversing the hash was performed and is discussed by (Murakami et al., 2010). “John the Ripper” the famous password cracking tool was modified in this experiment to use the CUDA architecture. In this experiment the dictionary was used and the number of threads was equal to the number of words in the dictionary. The cracking time was then compared with the dual core processor computer and was observed that CUDA took only 0.03% of the original time. Thus we can see that the famous hashing algorithms can be broken very easily now within no time due to the advancement in the hardware.

(Blocki et al., 2018a) compares the data breaches that took place in the giant companies like Dropbox, LastPass, Yahoo and Ashley Madison. These giant companies used the latest hashing algorithms like Bcrypt and PBKDF2-SHA256 algorithms. This paper challenges the security provided by these hashing algorithms. Also, it suggests that the use of the traditional key stretching algorithms like Bcrypt and PBKDF2-SHA256 should be avoided and the memory hard hashing functions like Scrypt or Argon2 should be used.

(Biryukov et al., 2016) compares the two famous memory hard hashing functions, i.e. Scrypt and Argon2. It says that the Scrypt algorithm calls many sub-procedures in a stack. It calls

SMix. SMix then calls ROMix, which calls BlockMix. BlockMix calls Salsa. The use of these subprocedures is not motivated. Also as they are called in stack, it impacts the time and space cost. Because of this stack structure the analysis of Scrypt becomes very difficult. The paper speaks about the common hashing problems that were highlighted by the Password Hashing Competition. It shows that how Argon2 overcomes these common problems. Hence from the study, we can conclude that, Argon2 is the safest algorithm available in market today.

## 2.4 Security of User Credentials

The hashed values are not secured to be stored in the database. Hence (George, n.d.) suggests storing the hashed passwords in the image using the technique of steganography. Steganography is the technique of hiding the on-going communication by hiding the information inside other information (Morkel et al., n.d.). The paper is classified into two phases: registration and login. In registration phase, the credentials in plain text are accepted from the user and stored in database except the hash. The password is then hashed and converted into a binary form. Convert the image into the binary format and hide the binary hashed password in the binary image. Then this image is stored in the secure location with the filename as email of the user. In login phase described by (George, n.d.), the credentials in the plain form are accepted by the user and the image is retrieved from the file. This image is then converted into binary form and the hidden binary hash is extracted from the image. This binary hash is then converted into the hash value and compared with the plain text password entered by the user. Hence we can see that this paper secures the hashes properly but it does not try to prevent the SQL injection attack. Also, the conversion of image into binary and then back to original and storing the files with the usernames will increase the space and time cost.

The similar approach of hiding the credentials is proposed by (Luo et al., 2019) with the use of negative password. This paper is also classified into two phases: registration phase and authentication phase. In registration phase, the user credentials are transmitted in the plain text form using secure channel from client to server. On server side the password is hashed and then converted into the negative password. The negative password is then encrypted and is stored in the database along with the username. In the login phase, the username and password is accepted from the client side and send to the server side. Depending upon the username, the corresponding negative password is obtained from the database and then decrypted with the key which is the hash of the plain password. Then the negative password is obtained. The password obtained from the user is hashed and compared to the negative password. If it matches, access is granted else denied. Here we can optimise this process further to achieve the same results by reducing the computation and hence the time required in this process.

The procedure used in (D'silva et al., 2017) during the registration phase, is by directly placing the credentials in the query. This query is then hashed using SHA1 algorithm and stored in the hash digest. During login phase, the user credentials are put in the query again

and the whole query is hashed using SHA1 algorithm. These hashes are then compared and the access is granted if the hashes match. The drawback in this paper is that it uses the very weak hashing algorithm which can be cracked easily. This procedure can be used for preventing the SQL injection attack. But as the hashing algorithm used is weak, it can be easily compromised and then user credentials along with the used SQL query will be visible in the database in the plain text format. Hence it needs additional security by improving the algorithm.

### **3 Research Methodology**

(Kindy and Pathan, 2011) and (Johari and Sharma, 2012) describes the need for preventing the SQLI attack. (Tatlı, 2015) explains how the hashes can be compromised easily and hence we should not store the hashes in the database. (Hongwei Wu et al., 2011) and (Murakami et al., 2010) explains how the advancement in hardware technology has reduced the time required to break the password. Hence we should use the improved hashing algorithms and also the hashes should not be stored in the database. (Blocki et al., 2018b) and (Biryukov et al., 2016) discusses the different hashing algorithms and suggests using the Argon2 hashing algorithm for hashing the passwords as it is the winner of Password Hashing Competition. Hence, in this project Argon2 is used to hash the user password and is taken care that it is not stored into the database.

(George, n.d.) shows us that how different techniques can be used for securing the credentials. The author in this paper has used the technique of steganography which means hiding the user credentials in the image and then storing the image in the file with the username of the user. However this process is time costly as we need to convert the image in the binary format and then convert the password in the binary image and then store it in the image. Also it is space costly, as we store the image corresponding to every user present in the database.

Hence, to overcome this drawback, (Luo et al., 2019) proposes the technique of calculating the negative password and encrypting it using AES algorithm. Still the process of calculating negative password and encrypting it seems to be a burden on the processor.

Hence from the above study, we realize that we should implement a method to secure the credentials by using secure hashing algorithm and encrypting that hash and storing it into the database. Hence, hashes are not stored in the database. Also there is no use of image steganography and hence time and space cost will be reduced. Also we are not performing extra operations like calculating the negative password. Hence this method of hashing the password with Argon2 algorithm and using AES encryption further is more optimised.

#### **3.1 Argon2 Algorithm**

Argon2 is the key derivation function. It was declared as the winner of the Password Hashing Competition held in July 2015 (Hodgkiss, 2018). (Wetzels, n.d.) highlights the advantages of



using Argon2. It shows that Argon2 has the common hash properties that are Collision Resistance, Pre-image Resistance and second pre-image resistance.

Argon2 also secures the hashes from lookup table attacks. In Argon2, similar messages produce different nonce. This means in order to crack the password, it is necessary to predict all the lookup tables for all the possible nonce values. The length of the nonce is 16 bytes. This means to crack the password, 2128 lookup tables need to be pre-computed and stored. The size of each lookup table is at least in several gigabytes. Hence it highly becomes resistant to lookup table attack. Argon2 provides us the defence against the optimised cracking of passwords because of its CPU and Memory Hardness nature.

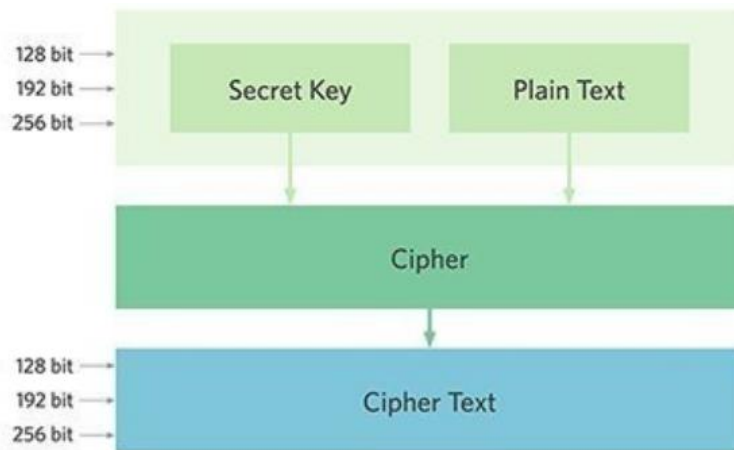
Argon2 has the following features (Biryukov et al., 2016):

- Performance
- Scalability
- Parallelism
- GPU unfriendly
- Trade-off resilience

Due to the above features of Argon2, it becomes the ideal candidate to be used in our application. (“argon2-cffi.pdf,” n.d.) describes that Argon2 generates the salt automatically and hence it hashes the same string with different salts generated automatically producing different output hashes. Also, it says that the verifier of the hash function is strong enough to verify if the given plain password is equivalent to the hash. This avoids the reason to store the salt in database.

## **3.2 AES Algorithm**

AES or Advanced Encryption System is the symmetric key algorithm. Symmetric key means same key is used for both encryption and decryption of the message (“What is Advanced Encryption Standard (AES)?,” n.d.). It is a block cipher that takes the input block of 128 bits. It has different sizes of keys: 128 bit, 192 bits and 256 bits. The size of the key is chosen depending on the security of the application and the cost of space and time. Depending upon the size of the key, the algorithm is classified into three types: AES-128, AES-192 and AES-256.



**Figure 1: AES Architecture**

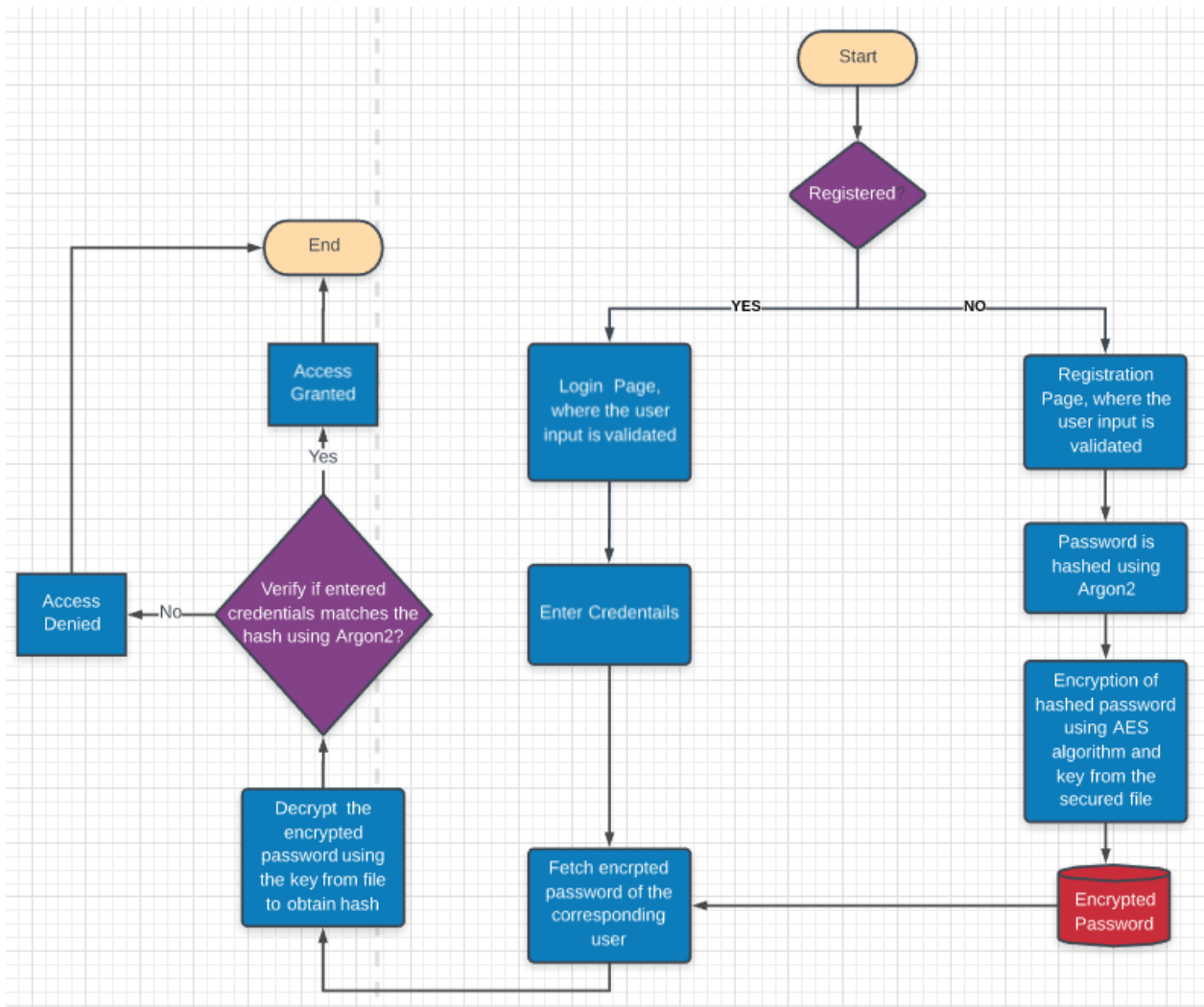
One round consists of substitution, transposition and mixing (“What is Advanced Encryption Standard (AES)?,” n.d.). Depending upon the size of the key the number of rounds is used. For example, if the key size is 128 bit, the number of rounds is 10, while the number of rounds increases to 12 and 14 for 192 and 256 bits respectively (“What is Advanced Encryption Standard (AES)?,” n.d.). In the system proposed, AES algorithm is used for encryption, because it is fast and secured (Mohurle and Panchbhai, 2016).

Also as said by (D’silva et al., 2017), we realise the need of preventing the SQL Injection attack itself. (“SQL Injection Prevention · OWASP Cheat Sheet Series,” n.d.) gives us the techniques that can be used for preventing SQL Injection attack which includes: using prepared statement, input validation, using stored procedure, avoiding user input and using least privilege method. These techniques are included in the implementation and hence the SQL Injection attack itself is avoided.

In this paper, a basic website is implemented. The evaluation is done by doing the SQL Injection attack on the website. Also, the response time of the website is calculated and discussed further in details in the evaluation section.

## **4 Design Specification**

This paper proposes a method, in which the web application is designed. This web application aims to be secured from the SQL Injection attack. Also, it tries to secure the passwords of the users by hashing it using the Argon2 hashing algorithm and then encrypting it using the AES encryption algorithm. The proposed model is classified into two stages: registration and authentication.



**Figure 2: Flowchart of the Proposed Algorithm**

#### 4.1 Registration Phase

1. In registration phase, the credentials of the users are taken through the registration phase.
2. Input by the user in each field is validated
3. Entered password by the user is hashed using Argon2 algorithm
4. Encrypt the hashed password using AES algorithm and the key used is obtained from the secured file
5. Store the encrypted password in the database

#### 4.2 Authentication Phase

1. In authentication phase, the user credential are taken through the login page
2. Input by the user in each field is validated
3. Depending upon the username entered, the corresponding encrypted password is fetched from the database.
4. Decrypt the password to obtain the hash using the same key from the secured file

5. Compare the obtained hash with the entered password using Argon2 verify function
6. If verification successful, access is granted, else access denied.

## 5 Implementation

This paper proposes the method, which is demonstrated by implementing a simple website for online pizza delivery. It has the table in the database that does not need to store the salt, because salt is generated and verified automatically by Argon2 algorithm as said in paper (“argon2-cffi.pdf,” n.d., p. 2). Table only stores the hashed and encrypted password itself. Table can be seen below:


idusers	UserName	Password	Role	UserState	timeStamp	Attempt	Address
31	Farhaan	LNzmv9AuL3S9x+VXRocG3pryJnnkEakDX1kHdKNps3P9...	Admin	Enabled	00:00:00	0	Pune
32	Suli	wQ6KenBQfrDykg9JufiGXaitUv37kZyj3F8IrPVMEvROw...	User	Enabled	00:00:00	0	Lucknow
33	Vishnu	vB7IYFILh5MJbxPAd7Jax00wppfShsHwZAs4dvNzgYMe...	User	Enabled	00:00:00	0	Kerala
34	Shairin	8uN/hshEVhCk4TEqBORl/wenfnEM71LVj5pSePGYJVgGL...	User	Enabled	00:00:00	0	Delhi
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 3: Design of Table in the Database

### 5.1 Registration Phase

The website has the registration page, using which the user can get registered to the website. This page verifies all the user input and by default the registration button is disabled. It also verifies the pattern given in the password.

Register with us... We'll never disappoint you.



Sign up here.....

Username   
Enter Valid User Name

Password   
Invalid Password!!!

Confirm Password   
password and confirm password not same

Address   
Enter Valid Address

+

have account [Log In](#)

Password must:

Figure 4: Registration Page Verifying User Input

If the user is already present, the registration button is disabled. If all the fields are proper, the registration button is enabled so that the user can register.

Sign up here.....

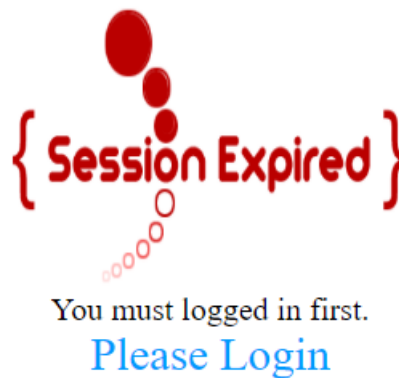
Username	Talha
Password	.....
Confirm Password	.....
Address	Mumbai

+

have account [Log In](#)

**Figure 5: Registration Page with Proper Entries**

On successful registration, the user is asked to login again.



**Figure 6: Web page after Successful Registration**

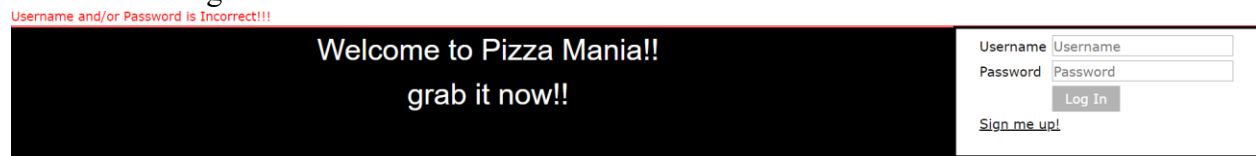
## 5.2 Authentication Phase

The website has the login page, using which the user can enter the username and the password. This page verifies the user input. If the user input is valid only then the login button is enabled and the user will be able to perform the login operation, else the button is not enabled.

Username	<javascript>
	Enter Valid Username
Password	.....
	Log In
	<a href="#">Sign me up!</a>

**Figure 7: Invalid Input is inserted in Login Page**

If the username entered is valid and does not exist, or if the password is not valid, it shows the error message as “Username and/or Password is Incorrect”.



**Figure 8: When Wrong Password is Entered**

On 3 unsuccessful login attempts, the user gets locked for 5 minutes and the following message is displayed “User Suspended. Try again after sometime!!!”. Depending upon the role assigned, the user gets the different homepage after logging in. Once the user is logged out, he cannot get in again by pressing the back button as the session is killed.

## 6 Evaluation

The website serves mainly two purposes. These purposes include the prevention of SQL Injection attack and the care is taken that in worst case, if in future the dataset gets compromised, the credentials of the user is safe. As we do a lot of tasks, it is also necessary to measure the response time of the application. In sub-section 6.1, the SQL Injection attack is done using OWASP ZAP tool. In sub-section 6.2, the SQL Injection attack is done manually. The response time is calculated in sub-section 6.3.

### 6.1 SQL Injection Attack using OWASP ZAP tool/ Case Study 1

Here, the tool named OWASP ZAP was used to attack to the website. This tool checks for all the OWASP top ten vulnerabilities. If any vulnerability is found it shows that with high risk. On attacking the website, there was no SQL Injection vulnerability found in the web application. Below, we can see the summary of the report of the attack done on the website.

Risk Level	Number of Alerts
<a href="#">High</a>	0
<a href="#">Medium</a>	1
<a href="#">Low</a>	3
<a href="#">Informational</a>	0

#### Alert Detail

Medium (Medium)	X-Frame-Options Header Not Set
-----------------	--------------------------------

**Figure 9: Summary of Attack Report**

### 6.2 SQL Injection Attack Manually / Case Study 2

The password is inserted which will always result true. The password entered was “Any\_02’ or ‘x’=’x””. As this will always result in tautology, it should be used for SQL Injection attack. But it fails with the following output in the screen.

Username and/or Password is Incorrect!!!

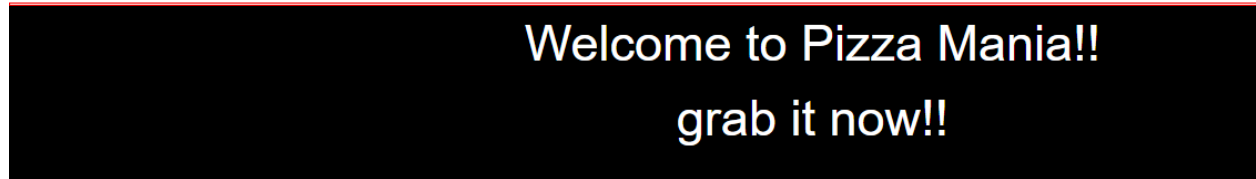


Figure 10: Failure of SQL Injection

The reason for this failure in SQL Injection attack, is the use of prepared statement.

### 6.3 Calculating Response Time using Apache Jmeter / Case Study 3

The response time or the throughput of the web application is tested. Here 2 threads, which correspond to 2 users are hitting the server simultaneously to achieve the response time of the server. It gives the throughput of 2.4/sec, which is normal.

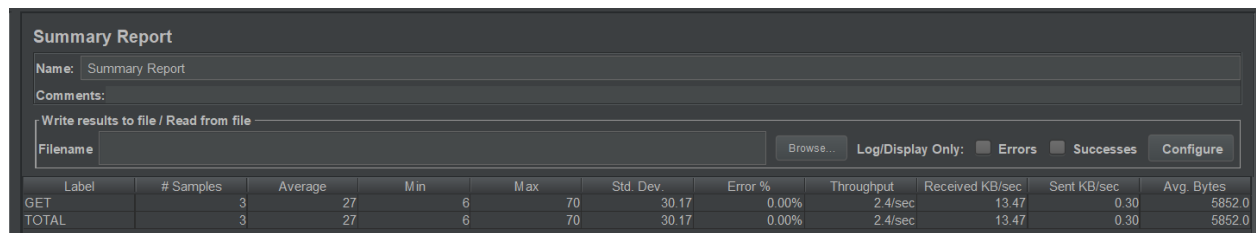


Figure 11: Result of JMeter

### 6.4 Comparison between Approaches to Secure User Credentials

Title of the Paper	Approach	Advantages	Drawbacks
Securing password hashes from SQL injection attacks using Image Steganography	After hashing of the password, it is not stored in the database and is hidden in the image	Hash is not stored in the database and hence even if SQL Injection attack takes place, user credentials are safe. Even if the attacker gets the image, he will not recognise that the hash is hidden in it, as the hash is hidden in the image using steganography.	The space cost is more, as it has to store a separate image for each user on the server. Also, time cost is more, as the operations are to be carried on the image.
Authentication by Encrypted Negative Password	The password is hashed and the negative password is encrypted and kept in the database.	Hash of the password itself is not stored in the database. If SQL injection takes place, the password of the user is never stored in the database.	Involves huge calculation as the negative password is calculated and then encrypted. No attempt to prevent the SQL injection attack.
Proposed Paper	The password is hashed and then encrypted. This encrypted password is	The hash is never stored in the database. There is no extra need to calculate the	Key is placed in the file. So if the key is compromised the

	then stored into the database.	negative password, hash itself is encrypted and kept in the database. Attempt is made to prevent the SQL Injection attack itself.	whole security of the database will be compromised.
--	--------------------------------	---	---

**Table 3: Comparison Between Approaches**

## 6.5 Limitations of Proposed Approach

In the proposed approach, the key is kept in the file. If the file comes in the wrong hands, it is very easy for the hacker to decrypt the password and then study the hash of the password and crack it. Still it needs to access the database to reach the encrypted password. This is highly impossible as the care is taken to prevent the SQL Injection attack.

The second limitation is that the paper focuses only on one of the SQL Injection attack. The other vulnerabilities of the OWASP top 10 are not considered in this paper. Hence even if the proposed method is implemented, there are chances of the compromise web application due to the other vulnerabilities present in it.

## 6.6 Discussion

A series of tests are performed here to test if the aim of the research was achieved. The aim of the research was to secure the user credentials form the SQL Injection Attack. For achieving this, the first test that was executed was the SQL Injection attack using OWASP ZAP tool. This test analysed the whole website to check if it included any vulnerabilities. It was not limited to only SQL Injection attack. In the result displayed above, we can see that there were total 4 vulnerabilities, out of which none was SQL Injection vulnerability. There was one medium level issue, which is X-Frame options header is not set. This is the limitation and this can be avoided, as our research was mainly focusing on SQL Injection attack. The thing to note in this experiment is that SQL Injection attack was not possible using the tool.

The second test case involves the SQL Injection attack that is done manually. Here the query is used, that will always result as true. Hence it was the Tautology type of SQL Injection attack. Care was taken to insert such a password that it does not break the rules of standard password. The standard password must contain at-least one number, at-least one small case letter, at-least one capital letter, the size of the password should be minimum 8 characters and maximum 20 characters. We see in the result that the error is thrown. The reason for this is the use of Prepared Statement and then binding the input with it.

The third test case is such that as we perform the operations like encryption and hashing algorithm. We see that the throughput looks good from the figure above. Brute force attack should not be possible if the throughput is too good. Hence we have used the slow hashing algorithm i.e. Argon2. Overall the evaluation is good only the need is to make the app more secure by putting the X-Frame Options header.



## 7 Conclusion and Future Work

Hence the aim of this research was to secure the credentials of the user stored in the database. We saw that the SQL Injection vulnerability ranks number 1 in OWASP top 10. Hence we did research in the field of securing the credentials in the database and found that storing the hash in the table is no longer secured. Hence in this research, the hash was not stored in the table. Also the additional security is required as the hash alone was not able to secure the password. Hence we did the AES encryption and stored hashed and encrypted password in the table. Moreover attempt was made to prevent the SQL Injection attack itself and was successful. This can be concluded from the experiments conducted to attack the database with the SQL Injection.

In future this paper can be extended to implement the security for preventing the application from the XSS attack. XSS attack stands for cross site scripting which injects the malicious code in the server. Using input validation technique this can be extended to prevent the XSS attack on the server.

## References

argon2-cffi.pdf, n.d.

Biryukov, A., Dinu, D., Khovratovich, D., 2016. Argon2: New Generation of Memory-Hard Functions for Password Hashing and Other Applications, in: 2016 IEEE European Symposium on Security and Privacy (EuroS P). Presented at the 2016 IEEE European Symposium on Security and Privacy (EuroS P), pp. 292–302.

<https://doi.org/10.1109/EuroSP.2016.31>

Blocki, J., Harsha, B., Zhou, S., 2018a. On the Economics of Offline Password Cracking, in: 2018 IEEE Symposium on Security and Privacy (SP). Presented at the 2018 IEEE Symposium on Security and Privacy (SP), pp. 853–871.

<https://doi.org/10.1109/SP.2018.00009>

Blocki, J., Harsha, B., Zhou, S., 2018b. On the Economics of Offline Password Cracking, in: 2018 IEEE Symposium on Security and Privacy (SP). Presented at the 2018 IEEE Symposium on Security and Privacy (SP), pp. 853–871.

<https://doi.org/10.1109/SP.2018.00009>

Capital One Data Theft Impacts 106M People — Krebs on Security, n.d. URL

<https://krebsonsecurity.com/2019/07/capital-one-data-theft-impacts-106m-people/> (accessed 8.9.19).

D'silva, K., Vanajakshi, J., Manjunath, K.N., Prabhu, S., 2017. An effective method for preventing SQL injection attack and session hijacking, in: 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT). Presented at the 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT), pp. 697–701. <https://doi.org/10.1109/RTEICT.2017.8256687>

George, T.R., n.d. Securing password hashes from SQL injection attacks using Image Steganography 16.

Hodgkiss, D., 2018. The Most Secure Password Hashing Algorithms. Damian Hodgkiss.

URL <https://damianhodgkiss.com/articles/most-secure-password-hashing-algorithms/> (accessed 8.9.19).

Hongwei Wu, Xiangnan Liu, Weibin Tang, 2011. A fast GPU-based implementation for MD5 hash reverse, in: Security and Identification 2011 IEEE International Conference on Anti-Counterfeiting. Presented at the Security and Identification 2011

- IEEE International Conference on Anti-Counterfeiting, pp. 13–16.  
<https://doi.org/10.1109/ASID.2011.5967405>
- Jake, 2017. Cracking everything with John the Ripper [WWW Document]. Medium. URL <https://bytesoverbombs.io/cracking-everything-with-john-the-ripper-d434f0f6dc1c> (accessed 8.9.19).
- Johari, R., Sharma, P., 2012. A Survey on Web Application Vulnerabilities (SQLIA, XSS) Exploitation and Security Engine for SQL Injection, in: 2012 International Conference on Communication Systems and Network Technologies. Presented at the 2012 International Conference on Communication Systems and Network Technologies, pp. 453–458. <https://doi.org/10.1109/CSNT.2012.104>
- Kindy, D.A., Pathan, A.K., 2011. A survey on SQL injection: Vulnerabilities, attacks, and prevention techniques, in: 2011 IEEE 15th International Symposium on Consumer Electronics (ISCE). Presented at the 2011 IEEE 15th International Symposium on Consumer Electronics (ISCE), pp. 468–471.  
<https://doi.org/10.1109/ISCE.2011.5973873>
- Kumar, H., Kumar, S., Joseph, R., Kumar, D., Singh, S.K.S., Kumar, P., Kumar, H., 2013. Rainbow table to crack password using MD5 hashing algorithm, in: 2013 IEEE Conference on Information Communication Technologies. Presented at the 2013 IEEE Conference on Information Communication Technologies, pp. 433–439.  
<https://doi.org/10.1109/CICT.2013.6558135>
- Luo, W., Hu, Y., Jiang, H., Wang, J., 2019. Authentication by Encrypted Negative Password. IEEE Transactions on Information Forensics and Security 14, 114–128.  
<https://doi.org/10.1109/TIFS.2018.2844854>
- Mohurle, M., Panchbhai, V.V., 2016. Review on realization of AES encryption and decryption with power and area optimization, in: 2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES). Presented at the 2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES), pp. 1–3.  
<https://doi.org/10.1109/ICPEICES.2016.7853276>
- Morkel, T., Eloff, J.H.P., Olivier, M.S., n.d. AN OVERVIEW OF IMAGE STEGANOGRAPHY 12.
- Murakami, T., Kasahara, R., Saito, T., 2010. An implementation and its evaluation of password cracking tool parallelized on GPGPU, in: 2010 10th International Symposium on Communications and Information Technologies. Presented at the 2010 10th International Symposium on Communications and Information Technologies, pp. 534–538. <https://doi.org/10.1109/ISCIT.2010.5665047>
- Rainbow table, 2019. . Wikipedia.
- Sajjadi, S.M.S., Tajalli Pour, B., 2013. Study of SQL Injection Attacks and Countermeasures. International Journal of Computer and Communication Engineering 539–542.  
<https://doi.org/10.7763/IJCCE.2013.V2.244>
- SQL Injection Prevention · OWASP Cheat Sheet Series [WWW Document], n.d. URL [https://cheatsheetseries.owasp.org/cheatsheets/SQL\\_Injection\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html) (accessed 8.10.19).
- Tatlı, E.İ., 2015. Cracking More Password Hashes With Patterns. IEEE Transactions on Information Forensics and Security 10, 1656–1665.  
<https://doi.org/10.1109/TIFS.2015.2422259>
- Top 10-2017 Top 10 - OWASP [WWW Document], n.d. URL [https://www.owasp.org/index.php/Top\\_10-2017\\_Top\\_10](https://www.owasp.org/index.php/Top_10-2017_Top_10) (accessed 8.9.19).
- Wetzels, J., n.d. The Password Hashing Competition and Argon2 17.

What is Advanced Encryption Standard (AES)? - Definition from WhatIs.com [WWW Document], n.d. . SearchSecurity. URL <https://searchsecurity.techtarget.com/definition/Advanced-Encryption-Standard> (accessed 8.10.19).