

Securing password hashes from SQL injection attacks using Image Steganography

MSc Internship
Cyber Security

Thomas Ronil George
x17121183

School of Computing
National College of Ireland

Supervisor: Mr Vikas Sahni

National College of Ireland
Project Submission Sheet – 2017/2018
School of Computing



Student Name:	Thomas Ronil George
Student ID:	x17121183
Programme:	Cyber Security
Year:	2018
Module:	MSc Internship
Lecturer:	Mr Vikas Sahni
Submission Due Date:	13/08/2018
Project Title:	Securing password hashes from SQL injection attacks using Image Steganography
Word Count:	4676 Words

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 encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature:	
Date:	17th September 2018

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
3. Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Securing password hashes from SQL injection attacks using Image Steganography

Thomas Ronil George
x17121183
MSc Internship in Cyber Security

17th September 2018

Abstract

SQL injection attacks can display hashed passwords on screen and it becomes accessible to unauthorized users. With access to the hashed password, it is not very difficult to identify the algorithm that was used to hash the password and decode it. This paper is intended for back-end or full stack developers who use SQL database systems to store hashed passwords in databases for user registration and login systems. The proposed method is to obfuscate password hashes in images using steganography, and to place them in a secure location rather than a database. These steganographic images will be used again for the login process. The key motivation for this paper is that a hash, when visible is quite intriguing and can help in learning patterns and algorithms. This paper demonstrates the successful implementation of a user registration and login system using image steganography for password storage.

1 Introduction

Securing sensitive information from hackers has been a major challenge in the field of cyber security and it is quite difficult to restrict the leakage of data in today's cyber environment. The reason for this is that it is impossible for every developer to know all the web vulnerabilities that are present. The vulnerability studied in this paper is the one that is ranked No. 1 by the OWASP (The Open Web Application Security Project) Top 10, 2017, which is SQL Injection - OWASP.org (2018). SQL Injections exploit applications that build SQL statements with user provided input. If the application does not validate the user input, then a hacker can manipulate the database with SQL commands. Bertino et al. (2007) describes several threat scenarios that arise when SQL commands are altered.

SQL injections are used by hackers to retrieve unauthorized information from databases. Some SQL based attacks display sensitive user information such as passwords or password hashes. The RockYou database hack was one such attack that hacked an entire database - Wikipedia.org (2018). The passwords from this database are being used to study password patterns even today. A study conducted by Narayanan and Shmatikov (2005) suggests that short words in the native language of users are easy to remember. Hence, a lot of users use short words in their passwords. Modern websites hence suggest

users to create passwords that include digits or special characters. This does not solve the issue with the security of passwords. The problem is that tools like John the Ripper use the same technique to generate dictionary words suffixed with digits and special characters. With these known facts, it is not very difficult for modern computers to reverse hashes of passwords. Rainbow tables are precomputed tables used to reverse hash values. Kumar et al. (2013) demonstrates a method to crack Windows password with a rainbow table. The paper also states that rainbow tables are used to crack password hashes in a short amount of time when compared to bruteforce, but the memory required is much higher. Hence, it is understood that if a hash is revealed it is possible to study it and reverse the value. A hash itself can reveal some information just by looking at it.

Prefix	\$1\$	\$2\$	\$sha-1\$	\$5\$	\$6\$
Hashing Algorithm	MD5	Bcrypt	SHA-1	SHA-256	SHA-512

Table 1: Prefixes of common hashing algorithms

The hashing algorithm used in this paper is bcrypt. The prefix '\$2a\$', '\$2b\$' or '\$2y\$' in a bcrypt hash string indicates that hash value belongs to a bcrypt hash function, in a crypt format. The rest of the hash string includes the cost parameter, a 128-bit salt (22 characters), and 184 bits of the resulting hash value (31 characters of encrypted output).

This paper focuses on securing passwords or password hashes from SQL Injection attacks through Image Steganography. Chandramouli and Memon (2001) states that the objective of image steganography is to communicate securely in an undetectable manner. Different digital formats can be used to implement steganography, but the formats that are suitable are the ones with high degree of redundancy. Anderson and Petitcolas (1998) define redundant bits as the bits which do not make a difference if they are altered. Hence, a text message can be embedded into a digital media such as an image and kept obfuscated. With this concept the password hashes can be stored in images. In case of a threat like SQL Injection on a database, this technique can ensure that the password hashes will not be compromised.

2 Related Work

In order to perform a SQL injection attack on the database, a study on SQL injection and the different types of SQL vulnerabilities are discussed in subsection 2.1. The passwords or password hashes from databases can help in studying password patterns. Subsection 2.2 contains techniques and methods used to analyze password patterns. By learning passwords and their patterns it is possible to crack passwords easily. However, this process is time consuming. But, studies show that with the development in technology it is possible to reduce the time required to crack a password. The implementation of this technique is discussed in subsection 2.3. Finally, subsection 2.4 discusses the different methods in image steganography.

2.1 SQL Injection

Structured Query Language Injection Attacks (SQLIA) is ranked 1st in the Open Web Application Security Project (OWASP) top 10 vulnerability list OWASP.org (2018). SQL injections can be done on websites to display unauthorized information on screen. By using the right manipulation code, it is possible to display user passwords or password hashes on screen from databases. Kindy and Pathan (2011) state that performing a SQL attack on a web application is equivalent to having access to the database of the web application. The data could be highly confidential. It could be the data of a hospital or a government institution. If hackers gain access to such data it could have fatal effects on its users and the organization. Hence, it is very crucial to protect a web application from a SQL injection attack. In order to have the perfect defence for a website against SQL injection attacks, it is important to know the types of SQL attacks and how they occur. Table 2 shows the different types of SQL attacks described by Kindy and Pathan (2011).

SQL Injection Method	How it occurs
Tautologies	One or more queries are prepared together in such a way that the result is always True.
Based on error messages	The database throws error messages if an SQL injection is coded incorrectly. The code is then manipulated until the SQL query is accepted.
Union concatenation	SQL queries can be combined together to collect data from other tables in the database with the keyword 'UNION'
Nested queries	Queries are added within queries to perform more actions through a single input.

Table 2: Types of SQL Injections

Johari and Sharma (2012) present a paper on detailed review of various types of Structured Query Language Injection attacks, their vulnerabilities, and prevention techniques. They also state that SQL attacks are one of the most common application layer attack techniques used by hackers to deface websites by manipulation or deletion of content using SQL queries. Their findings suggest that there are several mitigation techniques for SQL attacks, but all of them have their own weaknesses such as:

- Incomplete implementations
- Complex frameworks
- Runtime overheads
- Intensive manual work requirements

They conclude by stating that it is very important that the developers use good coding techniques for websites as they serve as the first line of defence.

2.2 Password Pattern Analysis

Studies have proven that in the past, the passwords from databases are compromised easily through various methods. Tatlı (2015) discusses some methods and techniques to understand the current security issues with passwords and databases. They state that a mistake that most developers make is to store user passwords within databases as plain-text or only hash values. There are many real-life scenarios where hackers have hacked user details from databases that include hashes or plain text passwords. After gaining password hashes of users, the hackers use brute force, dictionary or rainbow-table attacks to reverse the hash and ultimately find the plain text passwords. Tatlı (2015) worked with brute force attacks and improved them to compute hashes at a faster rate. The method targets password patterns that are commonly chosen by users. He suggests that pattern based brute force attacks are highly effective. To identify common password patterns, he performed both manual and automated analysis on the RockYou database. After identifying the patterns, he developed a generator, which generated many pattern-based passwords. He utilized the generated pattern-based dictionary to perform cracking tests against 15 different datasets. He was able to crack more password hashes than the RockYou database. This proves that it is possible to develop tools that can break hashes that were once considered to be unbreakable. Hence, password hashes must be made inaccessible.

2.3 Optimized Hash Reversal

With the improvements in the field of hardware technology, parallel processors have increased the computational speed of computers. A research conducted by Wu et al. (2011) optimized the reversal of MD5 (Message-Digest algorithm 5) hashes with the implementation of a GPU (Graphics Processing Unit) parallel architecture named CUDA (Compute Unified Device Architecture), which is a programming environment to develop GPU program code. CUDA comprises of a driver, a runtime library and a library for calculation. The driver performs the operations on the GPU hardware. The developed device was capable of running large giga threads in parallel. The design comprised of a module for message input traversing and another module for MD5 calculation. The MD5 algorithm was optimized by reducing the number of iterations in the MD5 transformation. The result was a 49 step MD5 function instead of 64 steps. Also, the GPU was optimized by syncing threads in reasonable positions. This experiment was 16 times faster than the CPU (Central Processing Unit). The paper concludes by stating that the method can also be applied to a wider range of one way hashing algorithms including SHA (Secure Hash Algorithm).

Murakami et al. (2010) conducted a similar experiment for reversing password hashes by modifying 'John the Ripper', which is a password cracking tool. The tool was used in parallel to compare hashes of words from a dictionary file with the help of CUDA. This method created a number of threads which were equal to the number of words in the dictionary file that was used. This technique reduced the hash execution time in the wordlist mode. The experiment was compared to the original John the Ripper process on a dual core. The result was that the processing time of the experiment was only 0.03% of the original time. This provides evidence that hash reversal methods are being improved. Hence, it is possible that the hashes that are considered secure today might not be very

secure in the near future. Thus, it is important to obfuscate hashes so that the chances of learning patterns from hashes can be avoided.

2.4 Steganography

Morkel et al. (2005) states that "Steganography is the art of hiding the fact that communication is taking place, by hiding information in other information". In order to obfuscate the password hash, this paper proposes a steganographic technique to embed password hashes in images. It is quite common to think of using only cryptography to secure information. In this case it is passwords. Wang and Wang (2004) suggest that cryptography is different from steganography. The key objective of cryptography is to keep important information as a secret. But, steganography is all about keeping the existence of the message as a secret. Cryptography is a good technique to keep communication secure but, the encrypted data that can be seen could be used to learn the encryption method that was used. This can be avoided if the existence of the secret message remains undetected. Thus, steganography is used as a key technique in this paper to hide the hash of the password. It is possible to apply steganography to most of the digital formats available such as text, images, audio, video and more. However, the only format used in this paper is the PNG image format.

Johnson and Jajodia (1998) define an image as the representation of a group of numbers in a computer. Each pixel has its own color value which is represented by 24 bits. The 24 bits are a representation of the primary colours: Red, Green and Blue. Each primary colour is represented by 8 bits. The three colours together constitute one pixel. Morkel et al. (2005) state that image steganography can be divided into two groups: image domain and transformation domain. The technique used in this paper is the LSB (Least Significant Bit) technique which falls under the image domain. The least significant bit is the last or the eighth bit of the eight bits that represent a primary colour. There are three LSB's in a pixel since three primary colours represent a pixel. The LSB substitution is a technique in which the LSB's are replaced with the binary value of the message to be hidden in the image. According to Morkel et al. (2005), a 800 x 600 pixel image can store 1,440,000 bits or 180,000 bytes of embedded information. The representation of a 24 bit image that has a 3 pixel grid is given below:

$$\begin{pmatrix} 01101100 & 10110011 & 10111010 \\ 01110001 & 00001110 & 11100010 \\ 11011010 & 10001101 & 10101011 \end{pmatrix}$$

The binary representation of the number 150 is 10010110. In order to perform steganography on the above pixel grid, each least significant bit is replaced by a bit from the binary representation of 300 as shown below:

$$\begin{pmatrix} 0110110\mathbf{1} & 1011001\mathbf{0} & 1011101\mathbf{0} \\ 0111000\mathbf{1} & 0000111\mathbf{0} & 1110001\mathbf{1} \\ 1101101\mathbf{1} & 1000110\mathbf{0} & 1010101\mathbf{1} \end{pmatrix}$$

3 Methodology

In order to demonstrate the security of the proposed model, two database tables have been created. The first table contains user information including the password hash just like every other website on the internet. The second table is similar to the first table and the only difference is that the password column is removed from the second table. The objective of developing the first table is to perform an SQL injection on it and prove that password hashes can be displayed on screen by manipulation of SQL code. The second table will be used to fetch user information except the password hash which is proposed to be stored in a steganographic image.

3.1 SQL Injection on a database table

Websites use a database system to store user information in tables. These tables usually contain usernames, passwords or password hashes, user id's, addresses and contact information.

A registration page handles the process of accepting user details, generating a hash for user password and storing user information in a database. The page captures username, email and a plaintext password entered by users and a hash is generated for the password. The function 'PASSWORD_DEFAULT' is used for generating the hash. The hashing mechanism used for the website is BCrypt. This is the default with PHP version 7.2.7. The collected user information and the password hash is stored in the first table. Figure 1 shows the structure of a traditional database table.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	userid	int(11)		No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/>	2	username	varchar(11)	latin1_swedish_ci	No	None			Change Drop More
<input type="checkbox"/>	3	password	varchar(256)	latin1_swedish_ci	No	None			Change Drop More
<input type="checkbox"/>	4	email	varchar(55)	latin1_swedish_ci	No	None			Change Drop More
<input type="checkbox"/>	5	active	int(1)		No	None			Change Drop More

Figure 1: Model of a traditional database.

A SQL injection is performed on the profile page of a user. The profile page consists of a SQL query to fetch some details of the user from users table and display them on screen. The code 'OR 1=1' is inserted at the end of the URL (Uniform Resource Locator) that contains the id of the user. The code breaks the SQL command and selects all the users in the table instead of just the user. Thus, the page will display the entire users table which also contains the hash of user passwords. A sample hash such as '\$2y\$10\$2QgGGVEsTv3wpM2czBAQPu.7Pr0bzi1DUDfV0tU.7D077DQHRf9IK', is used to understand the type of hash. This hash reveals that the hash is of 60 characters. The '\$2\$' tells us that the hashing mechanism used is BCrypt. '\$10' tells us that the cost parameter that was used is 10. What remains now is the salt and the hash. Thus, the hash can be broken down and studied.

3.2 Steganography with Python

Two files named 'enc.py' and 'dec.py' are developed to hide and retrieve messages from an image. The enc.py takes the password hash from the registration page and passes it as the message to a hide function. The hide function takes two inputs: the hash and an image. The image used is a RGBA (Red Green Blue Alpha) image named 'google.png'. The hide function first converts the hash into binary digits. Then it takes google.png and converts each pixel into its binary representation. Now the least significant bit in every pixel is replaced with one byte of the binary hash. Once the LSB substitution is complete, the new values are used to save the image. The image is saved in the name of the user email, making it unique. Thus the image saved is a steganographic image and it is saved in a secure location.

The dec.py uses the 'retrieve' function to retrieve the password hash from the steganographic image. The retrieve function takes only one input, that is the file name. The email of the user is passed as the filename. The retrieve function thus retrieves the steganographic image and converts it into its binary values. The function then returns the hash from the image.

3.3 Proposed model

The proposed method has a users table similar to the traditional users table. The difference is that the users table does not have a column for passwords or password hashes, as shown in Figure 2.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	userid	int(11)		No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/>	2	username	varchar(11) latin1_swedish_ci		No	None			Change Drop More
<input type="checkbox"/>	3	email	varchar(55) latin1_swedish_ci		No	None			Change Drop More
<input type="checkbox"/>	4	active	int(1)		No	None			Change Drop More

Figure 2: Structure of a proposed database.

The registration page takes the username and email and sends it to the database table. The next step is to pass the hash to the hashing function and get the password hash. Once the password hash is generated, it is now passed to the steganographic function to be embedded into an image. A sample RGBA image shown in Figure 3 is used for the steganographic function. Once the process is completed, the image with its new values is saved in the name of the respective user email taken as input. The image is stored in a secure location and will be used for the login process.

The login page takes the user email and a plaintext password as input. Then, the image that is saved in the name of the users email is accessed from the secure location. Once this is done the image is set as an input for the decryption process. The hash is retrieved from the steganographic image and sent to the 'PASSWORD_VERIFY' function to verify the plain text password entered by the user during registration. If the plain text password entered by the user for login matches the text extracted from the image, the user is allowed to login.

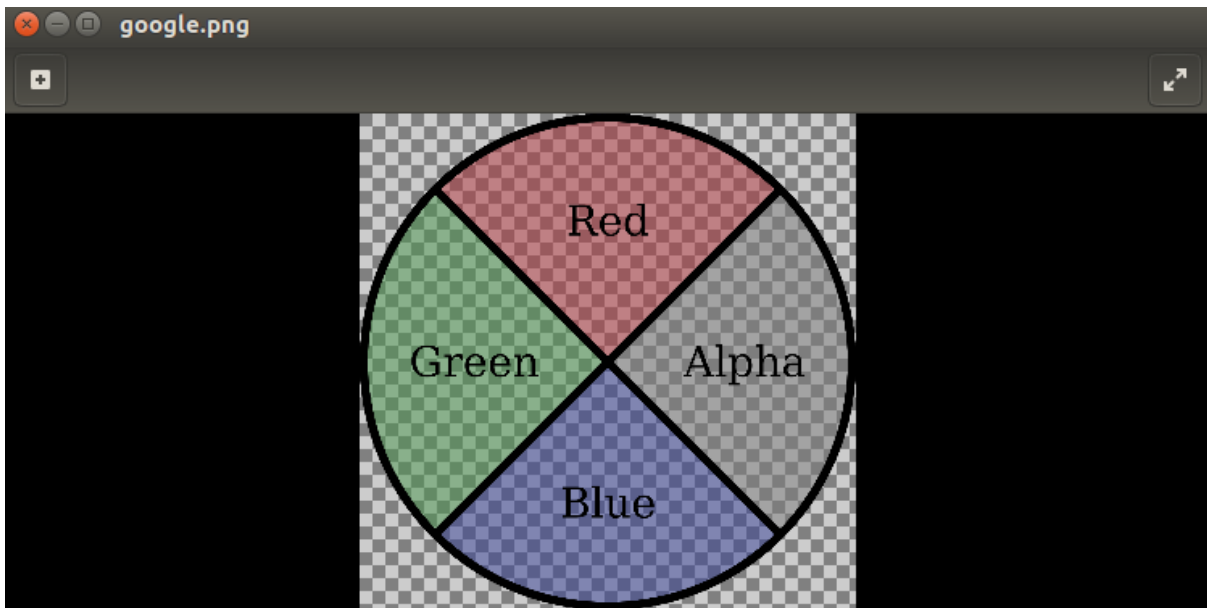


Figure 3: Image used for steganography.

3.4 Algorithm for registration/encryption process

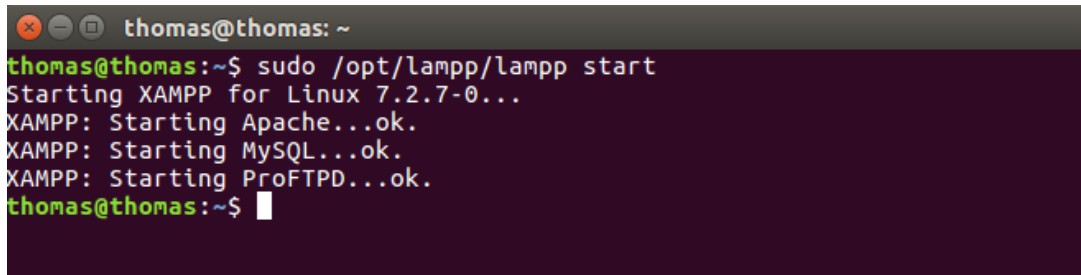
1. Capture the plaintext password and other fields entered by user
2. Generate a hash for the password with PASSWORD_DEFAULT function
3. Store the user information into a database table except the hash
4. Pass the hash to the python code to embed the hash into an image
5. Convert the hash to its binary value
6. Convert the image used for steganography to its binary values
7. Perform steganography with a hide function
8. Store the image in a secure location with the filename as the users email

3.5 Algorithm for login/decryption process

1. Capture the plaintext password and username entered by user
2. Retrieve the steganographic image from the secure location with the username provided
3. Convert the image to a binary format
4. Use a retrieval function to get the binary of the hash from the image
5. Convert the binary value of the hash to the original hash
6. Use the PASSWORD_VERIFY function in PHP to verify if the hash matches the plaintext entered by the user
7. If the password matches, allow the user to login

4 Implementation

The homepage of the website has two options. The first option uses a traditional method for user registration and login and the second option uses the proposed steganographic method. A local web server is set up and kept running. The web server used for the implementation is XAMPP which stands for Cross-Platform (X), Apache (A), MariaDB (M), PHP (P) and Perl (P). Apache and MySQL are two modules that are required for this step. These modules are used to run the website and to store user information in a database. Figure 4 shows the command to start XAMPP from it's directory.



```
thomas@thomas: ~  
thomas@thomas:~$ sudo /opt/lampp/lampp start  
Starting XAMPP for Linux 7.2.7-0...  
XAMPP: Starting Apache...ok.  
XAMPP: Starting MySQL...ok.  
XAMPP: Starting ProFTPD...ok.  
thomas@thomas:~$
```

Figure 4: Starting XAMPP server.

4.1 Traditional registration and login system

The traditional method has two buttons: first to register a user and second to allow a user to login. The register button leads to a registration page and the login button leads to the login page. The registration page contains a form that accepts user information and passes it to a database to be stored. During the process the password entered by the user is converted to a hash value. The stored information will be used to allow a user to login to the users profile. The login page accepts user email and password as input in a form. If the user details match, then the user is logged in to the profile page. The registration and login form are shown in Figure 5.

Once a user has logged in, the URL will contain the user id of the respective user at the end. Now, a SQL injection is performed to demonstrate the effect of the query. The query used to perform the injection is 'OR 1=1'. Figure 6 shows the manipulated URL.

Once the injection is done the result is that the entire table is displayed on screen as an output. The password hashes of users are now revealed. Figure 7 shows the user table displayed on screen.

4.2 Implemented steganographic registration and login system

The implemented method, which is the steganographic method also uses a similar interface for the registration and login process. The registration page of this method takes user details through a form and stores them in a users table except the password hash. The hash is sent to the python script to be embedded into an image. The image used for steganography is a sample image named 'google.png'. It will be used as one of

Register

Login

Figure 5: Registration and Login forms.



Figure 6: Adding SQL Injection code with URL.

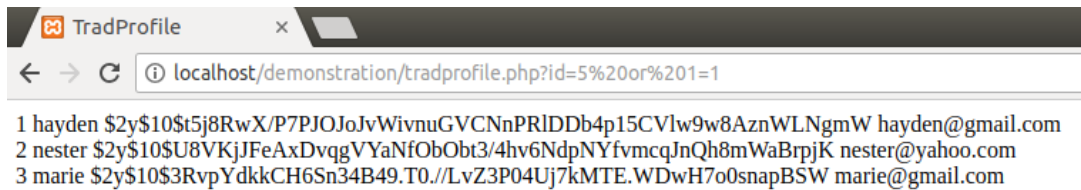


Figure 7: Output of SQL injection displayed on screen.

the inputs for the registration of every user. Once the steganography is complete, the image is saved with it's new values in a secure location as in Figure 8. The user email is used as the filename to save the new image. This will make the filenames unique.

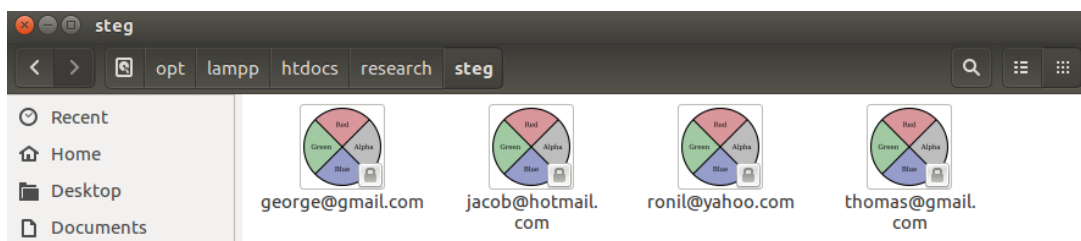


Figure 8: Generated Steganographic images.

Once the registration process is complete, the user is allowed to log in through the login page. The login page contains a form that will take user email and a plaintext password as inputs. Since the filenames of the steganographic images are user emails, the

email from the form is used to locate the steganographic image of the user. Once the file is found, it is sent to a python script to retrieve the hash from the image. The hash and the plaintext password entered by the user is given as input to the 'PASSWORD_VERIFY' function to check if the password is correct. If the password matches, then the user is logged into the profile page.

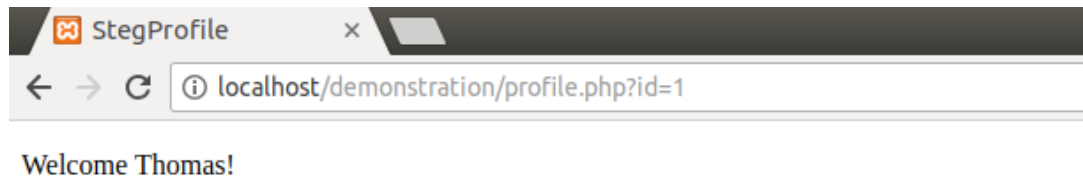


Figure 9: Profile page of logged in user.

Now, to check the output of a SQL injection the same code 'OR 1 = 1' is suffixed at the end of the url along with the user id. This time the output on screen does not have password hashes in the table. The hashes are secure in the steganographic images. Figure 10 shows the output of SQL injection on screen for the proposed method.

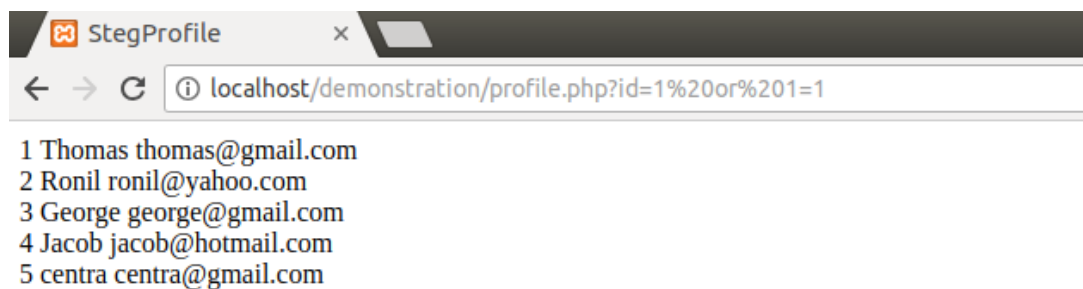


Figure 10: Output of SQL Injection on users table of proposed model.

5 Evaluation

In order to check the quality of the LSB substitution that was done, the image used before steganography is compared with the image generated after steganography in subsection 5.1. The outputs of SQL Injections are captured and discussed in subsection 5.2.

5.1 Before and after comparison of the image used for steganography / Case Study 1

Figure 11 shows the before and after pictures of the steganographic process. 'google.png' is the image before processing and 'thomas@gmail.com' is the image after the process. The steganographic image does not look altered with the changes made during the process. Hence, the images cannot reveal that they are steganographic images. This will ensure that the hashes are stored securely in the images and they can be used for user



Figure 11: Before and after comparison.

login and registration processes. In addition, these images are kept in a secure location that the public cannot access. In future, these images can be cryptographically locked to add security to the process.

5.2 The results of SQL Injections performed / Case Study 2

The database table used for the registration and login system of the traditional approach contain password hashes of all the users. Hence, when an SQL Injection is done on the regular users table, the output on screen contained a list of password hashes. Figure 12 shows the output of the SQL Injection for this method.

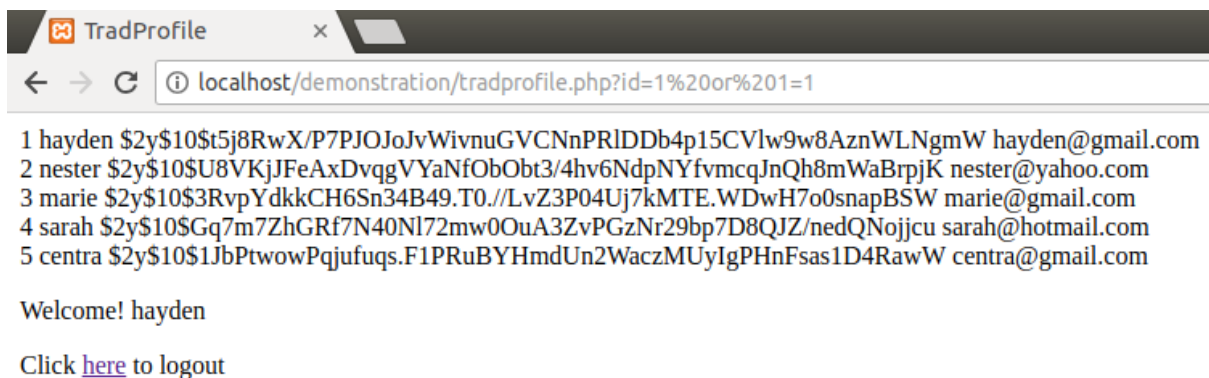


Figure 12: SQL injection on traditional database table.

The database table used for the registration and login system of the steganographic approach does not include password hashes. Hence, when an SQL Injection is done on the users table, the output on screen does not have any trace of hashes. Figure 13 shows the output of the SQL Injection for this method.

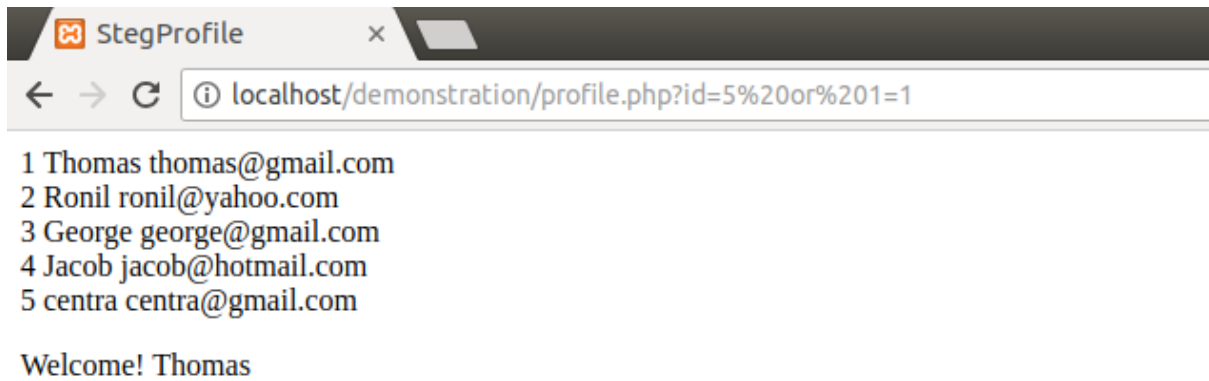


Figure 13: SQL injection on proposed database table.

5.3 Discussion

The proposed steganographic system was implemented and allows the registration and login system to function smoothly. The outputs of the SQL Injection prove that password hashes will be compromised if the right SQL command is used. Hence, it is possible that hackers can learn patterns from these hashes and build a hash reversing software in future. If the steganographic method is used for a user registration and login system, password hashes will remain obfuscated in steganographic images. The time taken for the method to complete a registration is 4 seconds longer than the traditional registration system. This is due to the time required to convert an entire image into its binary form for steganography. This can be optimized by making changes to process involved in steganography. An estimate of the number of pixels that is to be used for steganography should be made and only these pixels should be involved in the process. This will reduce the time required to complete the process of registration. Hence, this system can be used for user registration and login in future websites.

6 Conclusion and Future Work

A website with a traditional database table was created to study the impact of SQL injections. The SQL injection performed on the database table was able to manipulate the SQL command. The injection displayed the users table on screen including the password hashes. Hence, it is understood that password hashes are not entirely secure in database tables. The proposed method of embedding password hashes into images was successfully implemented and a user registration and login system was developed for the website. The technique provides additional security to password hashes.

Future work for this study could be to lock the steganographic images cryptographically. Passwords can be used to unlock the steganographic images. The LSB substitution used was a sequential substitution. The substitution method can be made random with true random number generators. This would ensure a randomized selection of pixels for Image Steganography and add more security to the image. Another future work would be to optimize the process of steganography by selecting only the required number of pixels for binary conversion instead of converting the entire image.

References

- Anderson, R. J. and Petitcolas, F. A. (1998). On the limits of steganography, *IEEE Journal on selected areas in communications* **16**(4): 474–481.
- Bertino, E., Kamra, A. and Early, J. P. (2007). Profiling database application to detect sql injection attacks, *Performance, Computing, and Communications Conference, 2007. IPCCC 2007. IEEE International*, IEEE, pp. 449–458.
- Chandramouli, R. and Memon, N. (2001). Analysis of lsb based image steganography techniques, *Image Processing, 2001. Proceedings. 2001 International Conference on*, Vol. 3, IEEE, pp. 1019–1022.
- Johari, R. and Sharma, P. (2012). A survey on web application vulnerabilities (sqlia, xss) exploitation and security engine for sql injection, *2012 International Conference on Communication Systems and Network Technologies*, IEEE, pp. 453–458.
- Johnson, N. F. and Jajodia, S. (1998). Exploring steganography: Seeing the unseen, *Computer* **31**(2).
- Kindy, D. A. and Pathan, A.-S. K. (2011). A survey on sql injection: Vulnerabilities, attacks, and prevention techniques, *Consumer Electronics (ISCE), 2011 IEEE 15th International Symposium on*, IEEE, pp. 468–471.
- Kumar, H., Kumar, S., Joseph, R., Kumar, D., Singh, S. K. S. and Kumar, P. (2013). Rainbow table to crack password using md5 hashing algorithm, *Information & Communication Technologies (ICT), 2013 IEEE Conference on*, IEEE, pp. 433–439.
- Morkel, T., Eloff, J. H. and Olivier, M. S. (2005). An overview of image steganography., *ISSA*, pp. 1–11.
- Murakami, T., Kasahara, R. and Saito, T. (2010). An implementation and its evaluation of password cracking tool parallelized on gpgpu, *Communications and Information Technologies (ISCIT), 2010 International Symposium on*, IEEE, pp. 534–538.
- Narayanan, A. and Shmatikov, V. (2005). Fast dictionary attacks on passwords using time-space tradeoff, *Proceedings of the 12th ACM conference on Computer and communications security*, ACM, pp. 364–372.
- OWASP.org (2018). Top 10-2017 top 10 - owasp, https://www.owasp.org/index.php/Top_10-2017_Top_10. (Accessed on 08/08/2018).
- Tath, E. I. (2015). Cracking more password hashes with patterns, *IEEE Transactions on Information Forensics and Security* **10**(8): 1656–1665.
- Wang, H. and Wang, S. (2004). Cyber warfare: steganography vs. steganalysis, *Communications of the ACM* **47**(10): 76–82.
- Wikipedia.org (2018). Rockyou - wikipedia, <https://en.wikipedia.org/wiki/RockYou>. (Accessed on 08/09/2018).
- Wu, H., Liu, X. and Tang, W. (2011). A fast gpu-based implementation for md5 hash reverse, *Anti-Counterfeiting, Security and Identification (ASID), 2011 IEEE International Conference on*, IEEE, pp. 13–16.