International Journal of Cyber-Security and Digital Forensics (IJCSDF) 7(4): 373-381
The Society of Digital Information and Wireless Communications (SDIWC), 2018 ISSN: 2305-001

# Analysis of Secure Hash Algorithm (SHA) 512 for Encryption Process on Web Based Application

Meiliana Sumagita[1] and Imam Riadi[2]
[1,2] Department of Information System, Universitas Ahmad Dahlan, Yogyakarta, Indonesia
Jl. Prof. Dr. Soepomo Sh, Warungboto, Umbulharjo, Kota Yogyakarta, Daerah Istimewa Yogyakarta
*(meiliana1300016043@webmail.uad.ac.id, imam.riadi@is.uad.ac.id)*

## ABSTRACT

The login mechanism in web-based applications implements the MD5 hash function as a form of password encryption that proves to have weaknesses in the form of Collision Attack that is likely to have the same hash value on two different input values that can threaten the security and confidentiality of the data. Implementation of password security in web-based applications requires updating to improve reliability and ensure system security by using SHA 512 method. Data collection was done by literature study, data collection from internet, and observation. The research method is divided into several processes, namely needs analysis and system vulnerability, and analysis for improvement. The program design consists of flowchart design and conceptual design of a hash function calling mechanism. Mitigation is carried out with the implementation of the new hash function calling a method, code change for system repair (patching) and test results from implementation. Testing is done by penetration testing and user acceptance test (UAT) Testing after application of patch, the inputted password has been converted to more reliable hash function using SHA 512 method, and the result of UAT shows the result agreed and strongly agree with 86, 00%, so the implementation of the patch used to secure the password that was made during login can run as required.

## KEYWORDS

Secure, Hash, Algorithm, Web, SHA 512.

## 1 INTRODUCTION

The security issues sparked the mechanism to control access to the network in order to protect it from intruders [1]. A vulnerability in a web application can be opening way for an attack in the whole information system and does not close the possibility for the control server [2]. One feature of the login feature is to authenticate users as identity checks where this function becomes an essential component of the security system. This is a way to differentiate between registered users and intruders. Authentication users on a network is a must for many companies that seriously protect their information assets and to know who and what will be accessed on their networks.

Various kinds of techniques for increase security data or information already is developed, one common way is by cryptographic or encryption techniques [3]. Use of encryption is needed to support the security of the login process. The sample web-based application used has been applied encryption method using Message Digest 5 (MD5) method so it must be updated using another more reliable method.

The SHA algorithm has a difference in the size of each block, the word of the data used during the hashing process, the length of the message can be processed, and the size of the resulting message digest varies according to the algorithm used, shown as in Table 1. The size of the Message Digest.

**Table 1.** Differences Each SHA Algorithm Variation

| Algorithm | Message Length (*bit*) | Block Size (in *bits*) | Word Size (in *bits*) | The Size of the Message Digest (*bit*) |
|---|---|---|---|---|
| SHA 1 | $<2^{64}$ | 512 | 32 | 160 |
| SHA 256 | $<2^{64}$ | 512 | 32 | 256 |
| SHA 384 | $<2^{128}$ | 1024 | 64 | 384 |
| SHA 512 | $<2^{128}$ | 1024 | 64 | 512 |

SHA 1 has a $2^{64}$-1 message input capacity, with 160 bits of hash results and $2^{80}$ hash power evaluations. Finally, in 2005 Rijmen and Oswald published an attack on the reduced SHA 1 version (using only 53 rounds from 80 rounds) and the results were found the collision with a complexity of about $2^{80}$ operations [4].

SHA 256 and 384 are not used much even though for security due to a protracted process that causes the length of time in hashing [4].
SHA 512 is a development of SHA 1 which is an MD4 based improvement. According to Megah Mulya [5], 2009, the reliability of SHA 512 is achieved by the ability to generate 512-bit hash values, which is the longest hash value that a hash function can generate. This long hash value makes the SHA 512 more resistant to attack than any other hash function so SHA 512 is considered a powerful, robust and fast hash function.

## 2 BASIC TEORY

### 2.1 Login System

An application or system that requires authentication of the owner of the access is definitely implementing the login system to secure the data. Login activity is generally done by entering data in the form of username and password that has been registered in the application or system. If the information is valid, then the client is allowed to access the network [6].

### 2.2 Encryption

Encryption is a process that changes a code from an understandable into a code that can not be understood or not readable. Encryption is intended to protect information from being seen by non-people or parties [7]. The way encryption works are shown in Figure 1.

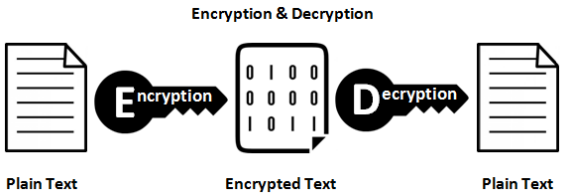



**Figure 1**. Working Mechanism of Encryption and Decryption

### 2.3 Hash Cryptography Algorithm

The one-way hash function is a one-way hash function [4]. A one-way hash function, also known as message summary or compression function is a mathematical function that takes the enter variable length and converts it into a binary sequence of a fixed length. The one-way hash function is designed in a way that is difficult to reverse the process, ie to find the circuit at a certain value (hence it is called one direction). The hash function is good if it is difficult to find 2 strings that will produce the same hash value [8]. The way the hash function works is shown in Figure 2.

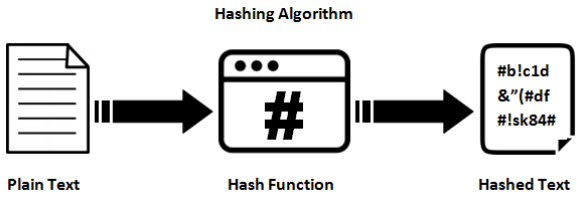



**Figure 2.** Working Mechanism of One Way Hash Function

### 2.4 *Secure Hash Algorithm* (SHA) 512

The SHA 512 algorithm is an algorithm that uses the one-way hash function created by Ron Rivest. This algorithm is the development of previous algorithms SHA 0, SHA 1, SHA 256 and SHA 384 algorithms. Journal of research Christian Angga [9], 2007, explains how the cryptographic algorithm of SHA 512 is receiving input in the form of messages of any size and generates message diggest which has 512-bit length.
Its predecessor is SHA1, and MD5 which is a renewal of MD4, the linkage, and development of the hash algorithm, indicating that the algorithm has proven to have been found to be a collision vulnerability. Currently, the National Institute of Standards and Technology (NIST) has made SHA 224, SHA 256, SHA 384, and SHA 512 as the new standard hash function. In Table 2 the resume parameters show some hash functions.

**Table 2.** Comparison of Multiple Hash Functions

| Algorithm | The Size of the Message Digest (bit) | Message Block Size | Collision |
|---|---|---|---|
| **MD2** | 128 | 128 | Yes |
| **MD4** | 128 | 512 | Almost |
| **MD5** | 128 | 512 | Yes |
| **RIPEMD** | 128 | 512 | Yes |
| **RIPEMD-128/256** | 128/256 | 512 | No |
| **RIPEMD-160/320** | 160/320 | 512 | No |
| **SHA-0** | 160 | 512 | Yes |
| **SHA-1** | 160 | 512 | There is a Disability |
| **SHA-256/224** | 256/224 | 512 | No |
| **SHA-512/384** | 512/384 | 1024 | No |
| **WHIRPOOL** | 512 | 512 | No |

SHA 512 hash function performs the same hash operation as SHA 2 operation in general [10]. SHA 512 hash function is a function that generates message diggest 512-bit size and 1024 bit block length. How the cryptographic algorithm works SHA 512 is to accept input in the form of a message with any length or size and will generate a message digest that has a fixed length of 512 bits as shown in Figure 3.

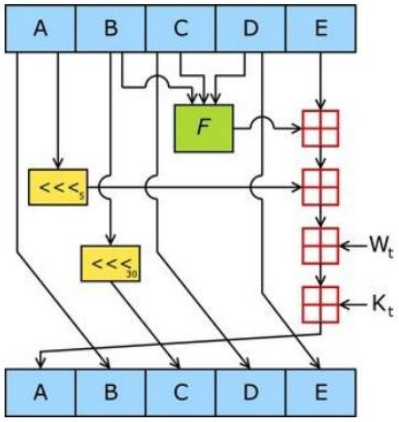



**Figure 3**. Working Illustration / Creation of Message Digest SHA 512

The workings of making message diggest with SHA 512 algorithm are as follows:

1. The addition of bits
   The first process is to add a message with a number of bit wedges such that the message length (in bits) is congruent with 890 mod 1024. The thing to remember is that the 1024 number appears because of the SHA 512 algorithm processes messages in blocks of 1024 sizes. If there is a message with a 24-bit length, then the message will still be added with the bundle bits. The message will be added with 896- (24 + 1) = 871 bits. So the length of the wedge bits is between 1 and 896. Then one more thing to note is that the bit bits consist of a bit 1 followed by the remaining bit 0.
2. Adding Long Message Redemption Value
   Then the next process is the message added again with 128 bits stating the length of the original message. If the message length is greater than 2128 then the length is taken in modulo 2128. In other words, if initially, the message length is equal to K bit, then 128 bit adds K modulo 2128, so after the second process is done then the message length now is 1024 bits.
3. Initialize Hash Value
   In the SHA 512 algorithm, the H hash value (0) consists of 8 words with 64 bits in the hexadecimal notation as in Table 3.

**Table 3.** Hexadecimal Notation SHA 512

| Buffer | Initial Value |
|---|---|
| A | 6a09e667f3bcc908 |
| B | bb67ae8584caa73b |
| C | 3c6ef372fe94f82b |
| D | a54ff53a5f1d36f1 |
| E | 510e527fade682d1 |
| F | 9b05688c2b3e6c1f |
| G | 1f83d9abfb41bd6b |
| H | 5be0cd19137e2179 |

## 3 METHODOLOGY

This section will explain the systematic way used to solve the research problem and also the steps undertaken in the testing and analysis of this research. The stages consist of literary studies is to analyze the system used to determine the current conditions, needs, advantages, and disadvantages of these programs. This stage is done by reading several books, previous research journals, papers or articles that are appropriate or relevant as well as collecting resources from the internet both journals, websites, proceedings and source code that can be used in this research.

Needs analysis and system vulnerabilities are carried out to analyze the vulnerabilities and needs of the system used. The analysis is focused on the web-based application login system encryption function which aims to find out the advantages and disadvantages of the encryption method currently used when replaced using the latest algorithm method.

Needs analysis and design for improvement are to describe and display an overview of the encryption process when the login is done. The description carried out is by showing a flowchart and conceptual diagram so that the work process in which password encryption is carried out until the login activity occurs can be delivered and understood more clearly.

Mitigation performed with the implementation of the latest hash function algorithm calling the method, code change for patching and test results from implementation. Testing in this study was conducted to show a comparison between the use of MD5 encryption method and SHA 512 encryption method. Testing was done

by Penetration Testing and User Acceptance Test. Penetration Testing is done by Brute Force testing while User Acceptance Test is done by filling out a questionnaire that is used as one of the recommendations to improve data security in web-based applications. These stages are described in Figure 4.

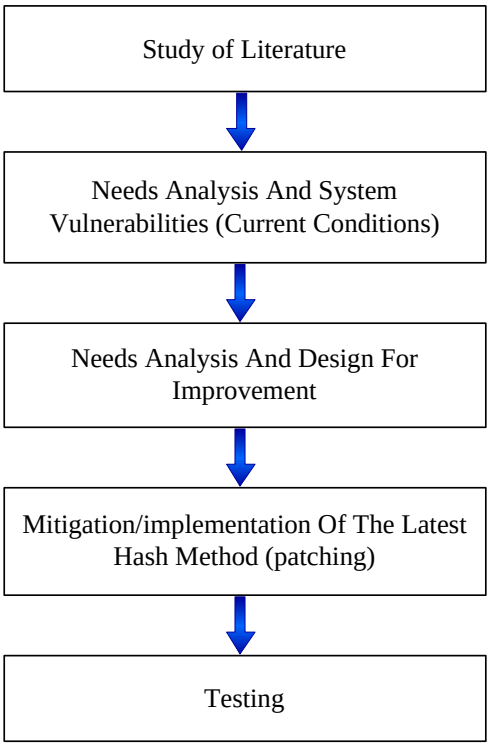



**Figure 4**. Block Diagram of Research Methodology

## 4 RESULTS AND DISCUSSION

### 4.1 Vulnerability Analysis

Information security is the preservation of information from all possible threats in an attempt to ensure or ensure business continuity, minimize business risk, and maximize or accelerate return on investment and business opportunitie [11]. This analysis is a discussion of how the system login on web-based applications running. This analysis is useful to know the vulnerability of the system so that it can know the improvements that need to be done.

This study discusses the analysis of encryption process in the web-based application using algorithm method of a secure hash algorithm (SHA) 512. The results of the analysis carried out will be used as a reference or alternative in managing web-based application login security systems. Analysis of the problem that is being discussed in this research will be explained in Figure 5.

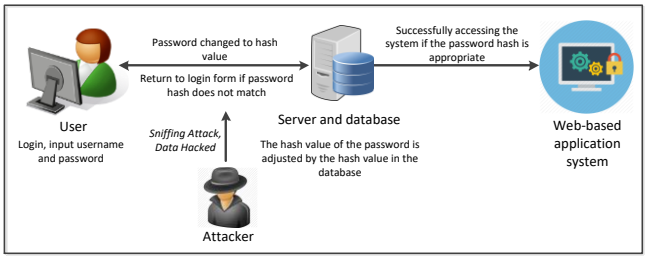



**Figure 5**. Schema Login In Web-Based Application

The explanation of the web-based application login scheme above is as follows.

1. Users enter the data in the form of username and password and send the data to the server. The data sent is data that has been changed using the MD5 hash.
2. The server receives the data in the form of a username and a hash value of the password that has been sent by the user.
3. The server will bring the hash value to the database to be equated with the hash value of the user's password. Verification process here is password verification in the form of hash value and not the password in plaintext form.
4. If the hash value sent by a user with stored in the same database then a user can enter and access system, but if not used will get a warning that the wrong password and return to main page login.
5. The problems found are the use of MD5 hash methods that are vulnerable to collision attacks [12], that threatening the security and confidentiality of data such as the MITM attack (Man In The Middle Attack) to conduct sniffing, spofing and other illegal activities [13].

### 4.2 Requirement and Improvement Analysis

This analysis is a discussion of what are the needs of improving the login system in web-based applications. After the needs and vulnerability analysis is done, the result is that the login system in the application must update the hash method used.

After knowing the hash method used in the login system, the use of encryption with the MD5 hash function has to be updated with more current and more reliable methods to maintain the security of an application or system. Renewal of this hash method is done by SHA 512 hash method which has more reliability than MD5.

### 4.3 Mitigation and Testing

#### 4.3.1 System Attack Scheme

The analysis process should be able to link information from different variable includes the completion of information against other information to explain an event or attacks activity [14]. Network forensics is defined in as capture, recording, and analysis of network events in order to discover the source of security attacks or other problem incidents. In other words, network forensics involves capturing, recording and analyzing of network traffic [15]. Sniffing on a computer network involves the use of a support tool that enables real-time monitoring. Sniffing in this study was done to check traffic on the network and retrieve a copy or capture of the packet data. The sniffing activity scheme is described in Figure 6.




**Figure 6**. Sniffing Activity Scheme

The sniffing experiment in this study was conducted with the Wireshark tool. Wireshark is one of the network packet analyzer tools. Wireshark will try to capture network packets and try to display the packet data as completely as possible. After the data obtained then will be analyzed the data capture results Wireshark to determine what type of hash function used by the system. The analysis to determine the type of hash function is done with the Hash Identifier tool. As an example of sniffing activities and analyzing the type of hash function performed in Figure 7 and Figure 8.

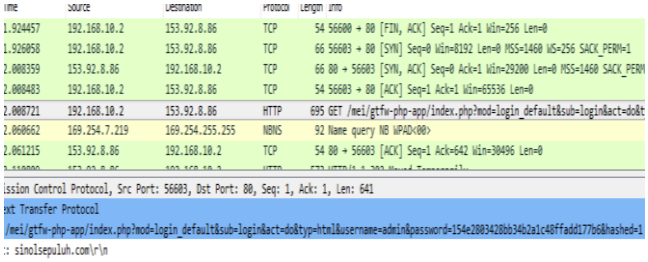



**Figure 7**. Sniffing Results Using Wireshark Applications

Figure 7 is the result of data traffic capture done using Wireshark tool. Capture data shows username information contains admin and password contains ciphertext with hash value 154e2803428bb34b2a1c48ffadd177b6. After obtaining the information is needed additional data that is the hash function is likely to be used by the system using Hash Identifier.




**Figure 8**. Hash Identifier results

Figure 8 shows that the login process on the application system has applied the MD5 hash method.

#### 4.3.2 Design Improvement

To facilitate analysis, a design and description for system improvement will be made by showing a flowchart and a conceptual image as described in Figure 9 and Figure 10.

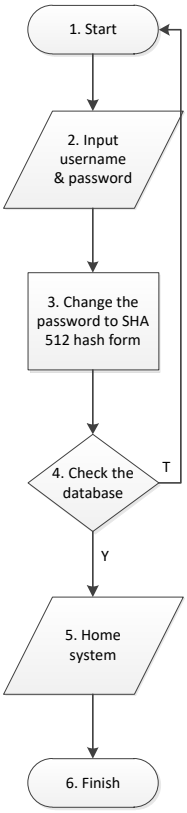



**Figure 9**. Flowchart Login Process Using SHA 512 Method

Based on the flowchart presented in Figure 9, process no. 3 that was previously encrypted using MD5 is changed using SHA 512 method. So in that process, the data transmission in the form of input from password will be changed to SHA 512 hash form which has the hash value much longer than MD5 therefore, user data will be more secure from a vulnerability that can occur when using MD5 as described in Figure 10.

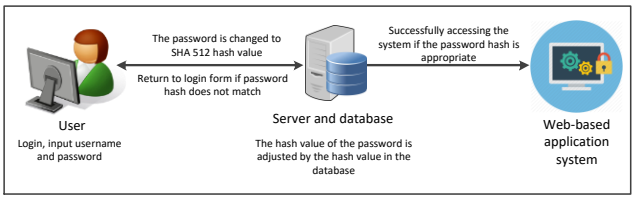



**Figure 10**. Image Conceptual Login Process Username and User Password Using SHA 512

The explanation of the conceptual image is as follows.

1. Users access the application and login to login to the application. The login process is done by sending data in the form of username and password. The process of sending data is done by changing the password data in the form of plaintext into SHA 512 ciphertext hash.
2. The application server receives the data in the form of the hash value of the password and then forward it to the database. This process is performed to verify the hash sent by the user whether it is the same as the password hash stored in the database (hash function for storing password).
3. If the data is suitable then the user can enter and access the application.

### 4.3.3 Patching Implementation

Renewal is done by changing the existing hash method into SHA 512 hash method combined with the addition of SALT secret key. Implementation done at this stage is encoding by creating a patch that will be used to call a hash function during login.

The plot of the calling process and the data changes for the username and password is first made before the encoding is done, so it can be known where the calling of the hash function calling can change the password to the ciphertext hash value. This process generates a flowchart calling the hash function on the system. There is also a diagram can be seen in Figure 11.

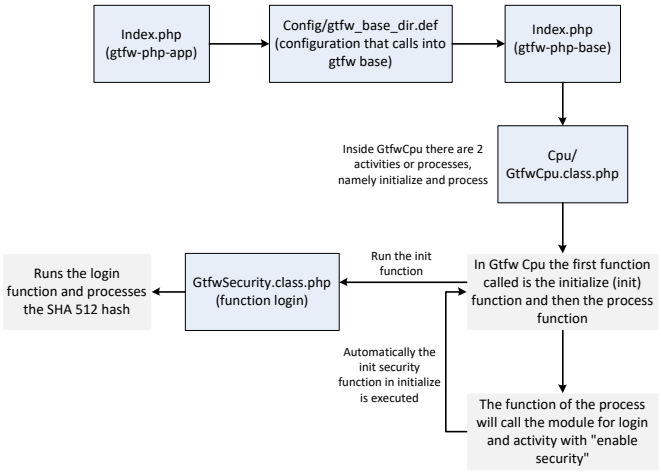



**Figure 11**. The Hash Function Calling Diagram

The process diagram shown in Figure 11 can be described as follows:

1. The index.php file is the first file executed by the program or application. This file will display the login form, will then call the gtfw-php-base file as the base library of the system.
2. The gtfw_base_dir.def file shows where the gtfw-php-base file is located and goes directly to index.php located in gtfw-php-base.
3. Inside index.php gtfw-php-base contains system libraries that invoke many functions or activities to process the system as a base/base function to run the system.
4. The index.php file on gtfw-php-base will process GTFW_BASE_DIR_CORE on the GtfwCpu.class.php file. The main functions that are processed are the initialize and process functions. The initialize (init) function is a function that will run the security command. The function of the process is a function that access/process some modules/actions, one of which is the module to log in. If in process function access login module and another module with "enable security" then automatically fungi init will be executed.
5. If init is done then will go to file GtfwSecurity.class.php and call the login function residing in it. This login function performs a hash method call for encryption to secure password data when login is done.

The coding done in this research includes changing the code or patch. The

GtfwSecurity.class.php file is used to print the hash value of a user-entered password by calling the SHA 512 hash function and receiving the hash value from the server to match the hash values stored in the database with the hash values generated from the input process. The source code line for the GtfwSecurity.class.php file can be seen in the script below.

```
$salt = $this->RequestSalt();
if ($hashed) {
$hash = md5(md5($salt .
$user['Password'])); } else {
$hash = $user['Password']; }
if (md5($password) === $hash) {
```

Changes and additions to code are done in the GtfwSecurity.class.php file, the addition of the code is used to call the SHA 512 hash function that has been implemented earlier in the GTFW application. The source code line for the GtfwSecurity.class.php file after adding the code can be seen in the script below.

```
$salt = $this->RequestSalt();
if ($hashed) {
$hash = hash('sha512', hash('sha512',
$salt . $user['Password']));
   } else {
$hash = $user['Password']; }
if   (hash('sha512',   $password)   ===
$hash) {
```

After the password is set to hash value, then the system will do the user data from the server then adjust the hash value with the existing in the database. Then stored in the login session and login process was successful.

### 4.3.4 Results of Patching Implementation

The result of adding code or scripts done in the previous process is to increase the value of security in the process of sending data. The result after the program is executed can be seen as in Figure 12 and Figure 13.

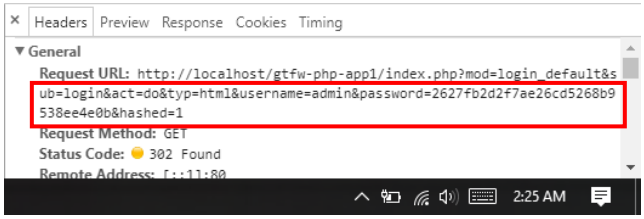



**Figure 12**. Results Process Call Hash Function Before Patching Performed

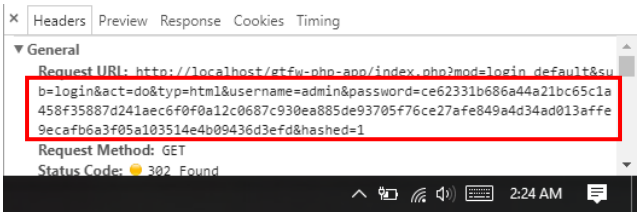



**Figure 13**. Results of the Hash Function Calling Process After Patching Performed

### 4.3.5 Testing

#### 4.3.5.1 Penetration Testing

This test has a purpose to show the resistance and strength of each algorithm against brute force attacks. The testing mechanism is to attack the resulting hash value by trying any combination to find the plaintext of the hash. This test is done by using Hashcat tool that serves to get plaintext from a hash or ciphertext. The result of this test is the comparison of time from which hash is faster-found plaintext him.
In the brute force test, the data obtained from the experiment is the time taken to obtain a plaintext that has been in the hash with MD5 takes an average of 54 seconds while the time taken for hash with SHA 512 takes an average of 68 seconds. Based on the test it was found that the SHA 512 algorithm is better in terms of durability and strength for brute force testing because it has a longer time to find the plaintext of the hash value of the algorithm.

#### 4.3.5.2 User Acceptance Test

User Acceptance Test is a testing process undertaken by the developer that will produce the document presented as evidence that the implementation of the program can be accepted by the developer in accordance with the required. The result of the percentage of user acceptance test is presented in a pie chart as shown below.

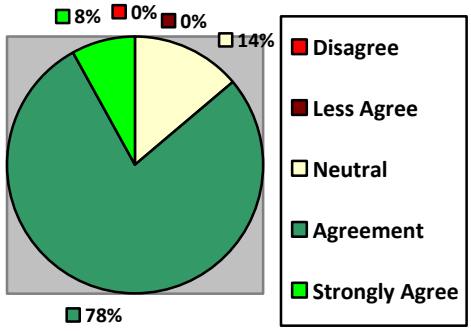



**Figure 14**. Percentage Test Result User Acceptance Test

Figure 14 shows the percentage of the above values indicates the responses of respondents to the statements in the Security Test questionnaire with SS answers of 8.00%, S of 78.00%, N of 14.00%, TS by 0.00%, and STS for 0.00%. Results obtained from the above test can be seen in Table 4 as table comparison of data security between before and after patching.

**Table 4.** Comparison Table Before and After Patching Performed

| No. | Parameter Comparison | Before Patching | After Patching |
|---|---|---|---|
| 1. | Security standard for login feature. | Not fulfilling, because it still uses the old hash method that has been proven to have a dangerous vulnerability. | Already fulfilled. The encryption update uses hash functions that have a more reliable and robust security level. |
| 2. | The level of password security on the mechanism of web-based application login feature. | Less good, because the algorithm method used has been proven to have dangerous vulnerabilities. | Good, because the algorithm method used proved more secure and reliable. |
| 3. | The total value of the hash function generated. | The resulting hash value is small so it only takes a while when a brute force test is performed. | The resulting hash value is much more so as to generate a long time when a brute force test is performed. |

## 5 CONCLUSION

Based on the results of research and discussion can be concluded that the login process in web-based applications requires updating of the encryption method used by the method of SHA 512 algorithm. This update aims to improve the security of password data on logging features that are more reliable and powerful so that the attacker will be very difficult to attack the system. Implementation of the SHA 512 algorithm method produces the longest number of bits of 512 bits so as to ensure system security and data confidentiality.

Penetration Testing against Brute Force attacks using the Hashcat tool indicates that the SHA 512 algorithm is better in terms of endurance and strength for brute force testing because it has a longer time to find the plaintext of the hash value of the algorithm thus indicating that the hash function is more reliable and robust. In addition to testing the User Acceptance Test generate agreed percentage and strongly agree at 86.00%, so the implementation of the patch used to secure passwords on the login feature can run as required.

## REFERENCES

[1] E. Kurniawan and I. Riadi, "Security level analysis of academic information systems based on standard ISO 27002:2003 using SSE-CMM," vol. 16, no. 1, pp. 139–147, 2018.

[2] I. Riadi, E. I. Aristianto, and A. Dahlan, "An Analysis of Vulnerability Web Against Attack Unrestricted Image File Upload," *Comput. Eng. Appl.*, vol. 5, no. 1, pp. 19–28, 2016.

[3] P. Irfan, Y. Prayudi, and I. Riadi, "Image Encryption using Combination of Chaotic System and Rivers Shamir Adleman ( RSA )," *Int. J. Comput. Appl.*, vol. 123, no. 6, pp. 11–16, 2015.

[4] M. H. W, "Development of Hash Function Encryption on SHA (Secure Hash Algorithm)," *J. Ilmu Komput. dan Teknol. Inf.*, vol. 3, no. 2, pp. 1–7, 2009.

[5] M. Megah, "Use of SHA-512 Algorithm to Ensure Integrity and Authenticity of Message on Intranet," no. 1, pp. 107–111, 2009.

[6] N. Hermaduanti and I. Riadi, "Automation framework for rogue access point mitigation in ieee 802.1X-based WLAN," *J. Theor. Appl. Inf. Technol.*, vol. 93, no. 2, pp. 287–296, 2016.

[7] A. Kristanto, *Data Security On Computer Networks*. Yogyakarta: Penerbit Gava Media, 2003.

[8] SSL Information, "Difference Between Hashing and Encryption," 2018. [Online]. Available: https://www.ssl2buy.com/wiki/difference-between-hashing-and-encryption.

[9] C. Angga, "Analysis of How Diverse Works Hash Functions Exist," pp. 1–6, 2011.

[10] W. Setiawan, "Analysis and Comparison of Whirlpool and SHA-512 Algorithms as a Hash Function," *Makal. IF3058 Kriptografi – Sem. II Tahun 2010/2011*, 2011.

[11] Y. P. Rosmiati, I. Riadi, "A Maturity Level Framework for Measurement of Information Security Performance," *Int. J. Comput. Appl.*, vol. 141, no. 8, pp. 975–8887, 2016.

[12] S. Dewantono, "Weakness of Message Digest

Function 5," 2011.

[13] M. S. Ahmad, I. Riadi, and Y. Prayudi, "Live Forensics Live From Investigation To Analyze Man Attacks in the Middle Attack Evil Twin Based," *Ilk. J. Ilm.*, vol. 9, no. April, pp. 1–8, 2017.

[14] M. I. Mazdadi, I. Riadi, and A. Luthfi, "Live Forensics on RouterOS using API Services to Investigate Network Attacks," *Int. J. Comput. Sci. Inf. Secur.*, vol. 15, no. 2, pp. 406–410, 2017.

[15] D. Mualfah and I. Riadi, "Network Forensics For Detecting Flooding Attack On Web Server," *IJCSIS) Int. J. Comput. Sci. Inf. Secur.*, vol. 15, no. 2, pp. 326–331, 2017.