The system of secured user's credentials transfer

Oleksandr Korchenko[1,*,†], Yevheniia Ivanchenko[2,†], Ihor Ivanchenko[2,†], Yevhenii Pedchenko[2,†] and Mari Petrovska[2,†]

[1] State University of Information and Communication Technologies, 7 Solomianska str, 03110 Kyiv, Ukraine
[2] National Aviation University, 1 Liubomyra Huzara ave., 03058 Kyiv, Ukraine

Abstract

This paper describes a developed system of secured user credentials transfer from a user's web browser to a web application server. This paper will describe in detail the operation of an advanced 10-level encryption algorithm for user credentials and demonstrate a High-Level Design system that represents the algorithm's step-by-step execution. The basis of this algorithm is the verification of user data to detect illegal (malicious) activity when entering a user's login and password. Based on the presented encryption algorithm, an experimental study of the developed user authorization web page and the processing of the entered credentials, before sending and after receiving the data processed by the web server.

Keywords

cybersecurity, MITM, brute-force, frontend, backend, authorization, authentication, user protection, database protection, data privacy, credentials

1. Introduction

Nowadays, where most companies work with clients and web applications, creating and using a personalized user account requires unique credentials, which are one of the key targets of attackers. For giant companies with a certain market share, developing a secure web application is a common task, as these companies can maintain a full department of developers who will develop and improve the web application, but for new companies or SMBs, this task is very critical, as they are unable to maintain a sufficient number of developers to cover all the necessary tasks for developing a web application, including security. Therefore, these companies either do not use security for their web applications at all and remain vulnerable to cyber-attacks or spend a significant portion of the company's budget on implementing 3rd party technologies to ensure the secure operation of web applications, secure user authentication, and transfer personal data.

Based on the analysis, Table 1 shows a list of the Top 5 commercial technologies used by companies and a list of threats that are prevented by these technologies.

2. Problems of transferring user's credentials between client and server

In modern cyberspace, where most of the world's users are located in virtual networks when companies migrating their computing to cloud service providers, and when the number of Internet users and unique devices is growing every day, the problem of ensuring secure users' credentials transfer to the web-application server remains.

Most modern web applications have a field for entering a login and password to gain access to the internal closed resources of the web application and the stored personal data of the credential holder. However, some of these web applications contain partial or do not contain security checks of the entered user's login and password, such as:

- Lack of security checks on a large number of password entries, which leads to a Bruteforce attack.
- Lack of security checks of the content of entered credentials by the user, which leads to a SQL-Injection attack etc.

In most cases, companies, in order not to modify the client-server part, which may affect the performance of the developed web application, use various commercial solutions to close and prevent potential cyber threats or cyberattacks, that have targeted the performance of the web application, leakage of commercial information and personal data of employees/customers of the company [1]. Therefore, the task of securely transferring user credentials entered in the form of a web application is relevant today.

Therefore, the main goal of this paper is to develop an effective and simple way to verify user input of credentials (login and password) to access the internal resources of a web application and prevent Bruteforce and Man-in-the-

CPITS-II 2024: Workshop on Cybersecurity Providing in Information and Telecommunication Systems II, October 26, 2024, Kyiv, Ukraine
* Corresponding author.
† These authors contributed equally.
agkorchenko@gmail.com (O. Korchenko);
evivancenko@gmail.com (Y. Ivanchenko);
ihor.ivanchenko@npp.nau.edu.ua (I. Ivanchenko);
pedchenko.ievhenii@npp.nau.edu.ua (Y. Pedchenko);
pmarisha2004@gmail.com (M. Petrovska)

0000-0003-3376-0631 (O. Korchenko);
0000-0003-3017-5752 (Y. Ivanchenko);
0000-0003-3415-9039 (I. Ivanchenko);
0000-0001-8436-5792 (Y. Pedchenko);
0009-0005-2150-2194 (M. Petrovska)

Middle (MITM) attacks using improved and built-in web application tools [2].

To achieve this goal, the following tasks need to be completed:

1. Analyze existing technologies and web attacks that cause monetary and reputational losses to companies.
2. Develop an improved HLD (High-Level Design) system, algorithm, and software solution for secure transmission and verification of user credentials (login and password).
3. Conduct an experimental study of the developed improved version of the user credentials input verification.

The novelty of this paper is the improvement of the user authentication system on web applications by applying procedures for security verifying the input of credentials and contents to detect malicious code, which will allow companies to protect the transfer of credentials and personal data of users at the initial stage of the company's growth and protects company's services from leakage, hacking and disruption without the use of third-party technologies [3].

Based on the analysis, Table 1 shows a list of the Top 5 commercial technologies used by companies and a list of threats that are prevented by these technologies:

Table 1
Top 5 commercial technologies that are needed to protect companies' web application

Technology|Attack	Bruteforce Attacks	OWASP Top 10 Attacks	DoS Attacks	DDoS Attacks	Bots Attacks	Malware Attacks	Man-in-the-Middle (MITM) Attacks
Web Application Firewall (WAF) [4]	+	+	+		+	+	
DoS/DDoS Protection [5]	+		+	+			
Multi-factor Authentication (MFA) [6]	+						+
Static Application Security Testing (SAST) [7]		+				+	+
Dynamic Application Security Testing (DAST) [8]		+			+	+	+

It should be considered that companies eliminate vulnerabilities that are found after writing web applications and publishing them on the Internet using each of the above technologies.

3. Developing of high-level design system

To present the principle of operation of the encryption methods used to transfer the login and password to the web application server, a High-Level Design (system) was developed that visually demonstrates the step-by-step processing of the input data by the user and their sequential transmission to the web application server.

This approach is based on the 10-level principle (each of these levels will be described below) of transferring login and password to authenticate the user and ensure the authorization of a legitimate user to the internal resources of the web application, as shown in Fig. 1.

Level 0. This level is used to establish an encrypted connection between the client and the server, in this case between the client's browser and the server side of the website using the SSL protocol, which contains a set of ciphers and protocols not lower than TLS 1.2 or TLS 1.3 [9], and the size of the encryption key is not lower than 4096 bits. The use of this encrypted connection is necessary to avoid MITM attacks and to prevent a legitimate user from joining the session.

Level 1. At the first level, after successfully establishing a connection with the web server, the user is displayed the web application login page, where the user is prompted to enter the login and password of the user registered in the web application.

Level 2. After the user enters the login and password, a Hash function is generated for the login using the SHA-256 encryption algorithm [10] to make it impossible to decrypt the encrypted data and transfer it to the web application server for further processing.

Level 3: After the web application server receives the encrypted user login, three additional protection and verification modules are enabled to prevent the use of the Bruteforce attack.

- The first stage is the store of the client's IP address.
- In the second stage, the number of unsuccessful attempts to authorize the client is counted.
- In the third stage, the received content is checked for embedded injections that may affect the functionality of the web server and the web application database.

After successful verification by the built-in security modules and a verdict that the session is from a legitimate user, the main module is connected, which downloads all possible Hash functions of all users' logins to the web server's RAM, and searches this list for the received Hash login from the user.

Level 4. After finding an identical unique user, a one-time "Salt" (token) [11] is created on the web server side, which is recorded in the database and transferred to the user. This token is valid only once and the next time the user logs in, the "Salt" will be change**Level 5**. When the client receives a one-time token, the password encryption process

starts. To transfer the password, we use our own developed encryption algorithm consisting of three steps:
Creating a Hash function of the password entered by the user.

Adding a one-time "Salt" (token) to the created Hash function of the password.

Creating a Hash function for the data string obtained in the second step.

As a result, we get an encrypted password that cannot be decrypted back, since a one-way encryption method is used.

Also, it is important to note that the encrypted password string will be different for each authorization, so this helps to avoid MITM attacks and prevent the reuse of the encrypted password string for authorization on the web application.

Level 6. After encrypting the user's password at Level 5, the encrypted login and password are transmitted to the web application server via the created SSL channel.

Level 7. After the web application server receives the encrypted login and password from the user, the procedure described in Level 3 is repeated, and the user's login is compared by the Hash function, which allows you to get a one-time token (Salt) and password in the form of a Hash function from the database, and a similar operation is performed as described in Level 5 for the received encrypted data. After all the necessary calculations are performed, the generated encrypted password is compared with the one-time token and the encrypted password received from the client.

It is important to note that before comparing the data, the data received from the user is checked for injections to prevent database hacking and disruption of the web server.

Level 8. At this almost final level, in case of successful comparison of the login Hash function and the encrypted password received from the user, a final one-time access token is generated for the user, which entitles the user to access the internal resources of the web application and receive a link to the main page. Accordingly, the new one-time token is also stored in the database.

Level 9. After successfully passing all of the above levels, the user successfully authenticates to the web application and follows the link received from the web application, which gives the user the right to use the capabilities of the web application within the limits of the authorized access rights and the configured role-based access model (RBAC) [12].

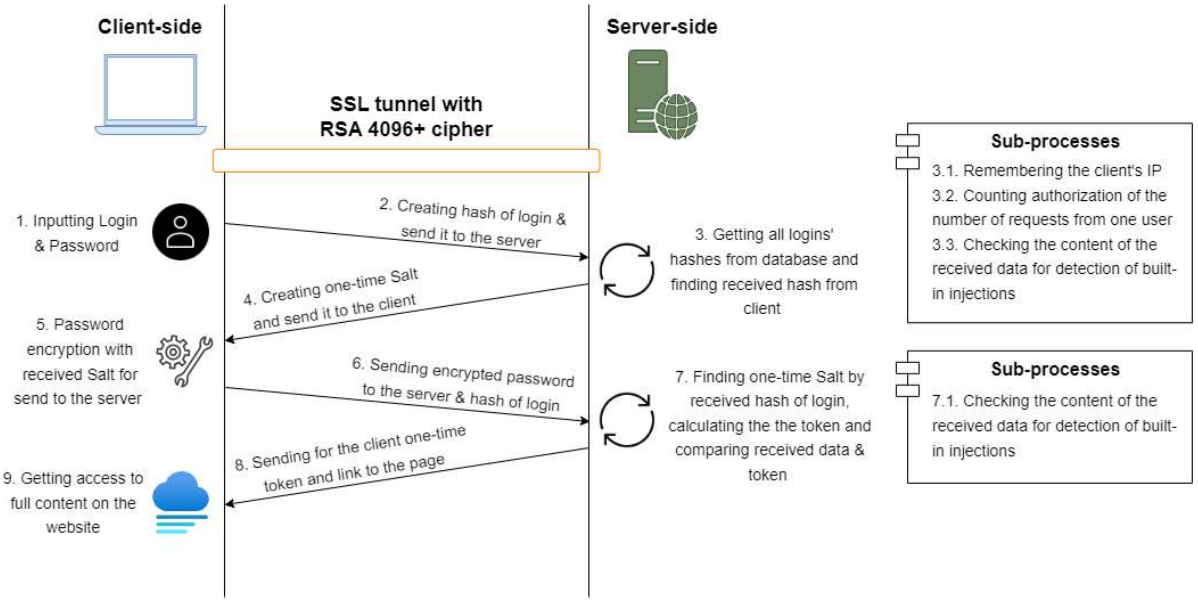

Figure 1: High-level design system of secure login and password transmission from client to web application server via secured tunnel

The algorithm described above, presented as an HLD system, will allow companies developing web applications to provide comprehensive verification and secure transfer of user credentials to the webserver to avoid the possibility of MITM and Bruteforce attacks and prevent the need to purchase commercial solutions (types of which are presented in Table 1) at the early stages of company and web application development.

4. Experimental investigation of the developed algorithm processing for secure transmission of user's credentials

Based on the third section of this paper, which presents the algorithm of the developed approach to securely transfer user credentials to the web application server, the practical application of this algorithm will be presented. Demonstrations of the developed algorithm will be presented below in this section.

For example, it will be used in the web application under development—"Enco Console", which contains a user login and password field to access the internal capabilities and data of the web application (Fig. 2). For testing, we used the next test credentials: root (login) and qwerty12345 (password).

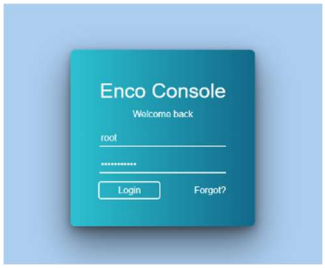

Figure 2: User authorization form

When you enter the user credentials and click the "Login" button, the algorithm described in the third section of this paper is executed. To provide a visual view of how user data is being processed at each stage of execution, a function was added that displays an intermediate result.

At level 2 of the developed algorithm, after entering the login and password, a hash function of the user's login is created and the processed data is sent to the web server (Fig. 3), as shown in the image below.

After successfully transferring the user's login Hash function to the web application server and performing the necessary checks on the server side, the user receives a one-time token that will be valid only until the user is re-requested to log in (Fig. 4).

```
root
qwerty12345
4813494d137e1631bba301d5acab6e7bb7aa74ce1185d456565ef51d737677b2
```

Figure 3: Creating a Hash function for the user's login "root"

```
root
qwerty12345
4813494d137e1631bba301d5acab6e7bb7aa74ce1185d456565ef51d737677b2 <= Login Hash
a346d3e4718069c1ef3c9f0ec317d470a576ac80527531a4c4e80de306473ce0 <= Salt (one-time token)
```

Figure 4: Successfully receiving a one-time token (Salt) from the web server

After the successful receipt of a one-time token on the client side, the process of encrypting the user's password takes place, which must be transferred to the webserver to gain access to the internal resources of the web application (Fig. 5). There are three main operations involved in encrypting a user's password:

1. Creating a Hash function for the entered user password makes it impossible to decrypt this line of code.
2. Adding the received one-time token (Salt) to the created password Hash function and forming a single 128-character expression [13].
3. Create a Hash function for the single expression created in step 2.

```
root
qwerty12345
4813494d137e1631bba301d5acab6e7bb7aa74ce1185d456565ef51d737677b2
a346d3e4718069c1ef3c9f0ec317d470a576ac80527531a4c4e80de306473ce0
f6ee94ecb014f74f887b9dcc52daecf73ab3e3333320cadd98bcb59d895c52f5 <= Password Hash
f6ee94ecb014f74f887b9dcc52daecf73ab3e3333320cadd98bcb59d895c52f5a346d3e4718069c1ef3c9f0ec317d470a576ac80527531a4
c4e80de306473ce0 <= Password Hash + Salt
a8911cb41e1369b94315e9cab9a0ec5182d0d8a2c007a763a62110ab48a59aaf <= (Password Hash + Salt ) Hash
```

Figure 5: Encrypting a user's password before sending it to the web application server

After the web application server has successfully verified the received function, the user is sent a session key that will be valid only for one user session and, at the next login, both the one-time token and the one-time session key will be changed. Along with receiving the session key, the user receives a link to a page that provides access to the internal resources of the web application (Fig. 6).

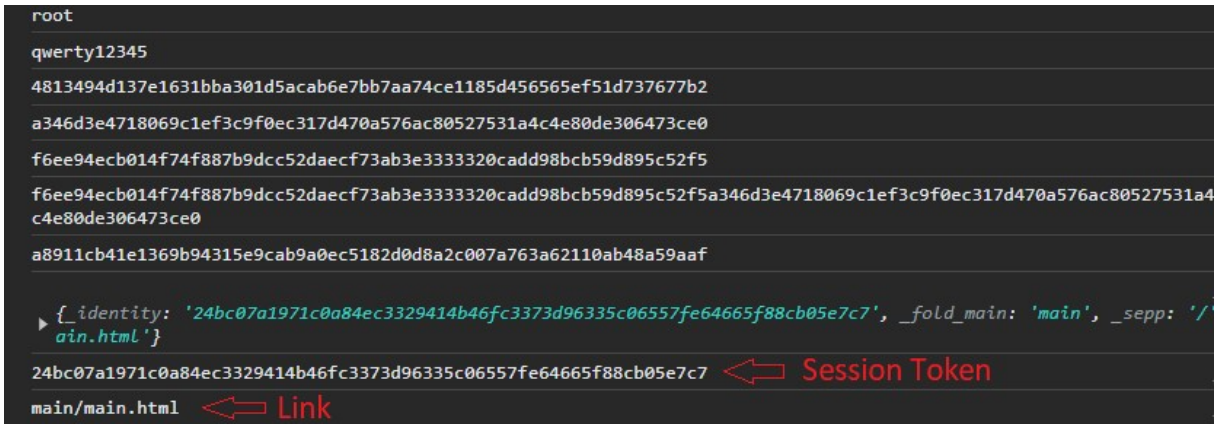

Figure 6: Getting a session key and a link to go to the next internal page of the web application

As you can see, the developed algorithm has 10 levels of verification, where each level launches and executes a different set of procedures that check each field entered by the user and the data transmitted to the web application server. That's why, thanks to the implementation of this algorithm, businesses in the early stages of development have the opportunity to protect their resources from two types of attacks: MITM and Bruteforce since to prevent MITM attacks, it is assumed that it will not be possible to resend the token, since when the authorization page reloads, the token will already be updated and, at the same time, the data received from the user does not interact with the database directly, but only through the server side of the web application. To prevent a Bruteforce attack, the server side, when receiving hash functions, logs the number of successful/failed authorization attempts by the user, where it is established that after 3 unsuccessful password attempts, the user's account and IP address are blocked for up to 30 minutes.

5. Conclusions

The analysis conducted at the beginning of this paper has identified the key needs and threats that require the purchase of commercial web application security products. For enterprise-level companies, these solutions are available, as well as the engineering resources that will be involved in supporting their operation, but for small SMB companies or new companies that are just starting, the purchase of the described set of solutions is unaffordable. That is why it was decided to improve the existing algorithms, which will allow new companies or SMBs to protect the credentials of users or customers of a web application from attacks such as Man-in-the-Middle and Bruteforce. To build a secure data transmission channel between the client and the server, an algorithm consisting of 10 levels was proposed, which describes the interaction between the client's web browser and the web application server and describes the step-by-step operation of the encryption algorithm. Also, while using and implementing this algorithm on the authorization page, it is worth considering subprocesses that are connected at two levels on the server side and provide additional verification of user activity when entering a login and password on the authorization page. Also, for visualization, the last section provides a practical application and display of the algorithm at each stage of the transmission and receipt of encrypted data between the client's web browser and the web application server, which allows you to clearly understand what data is being processed and at what time. Through the use of one-time tokens, the company can avoid reusing tokens and compromising data in the database by gaining unauthorized access to the data in the database. Also, it is worth noting that this algorithm works in such a way that the data sent to the web server is not transferred to the database in any way, and the data is processed on an intermediate server that interacts with both the client and the database, which prevents SQL-Injection attacks and the use of any other injection that relates specifically to the corruption of data in the database.

References

[1] Y. Shcheblanin, et al., Research of Authentication Methods in Mobile Applications, in: Cybersecurity Providing in Information and Telecommunication Systems Vol. 3421. (2023) 266–271.

[2] M. TajDini, V. Sokolov, V. Buriachok, Men-in-the-Middle Attack Simulation on Low Energy Wireless Devices using Software Define Radio, in: 8th International Conference on "Mathematics. Information Technologies. Education": Modern Machine Learning Technologies and Data Science, vol. 2386 (2019) 287–296.

[3] D. Shevchuk, et al., Designing Secured Services for Authentication, Authorization, and Accounting of Users, in: Cybersecurity Providing in Information and Telecommunication Systems II Vol. 3550 (2023) 217–225.

[4] Akamai's Team, What Is a Web Application Firewall (WAF)? (2023). URL: https://www.akamai.com/glossary/what-is-a-waf

[5] AWS's Team, What is a DDoS Attack? (2023). URL: https://aws.amazon.com/shield/ddos-attack-protection/?nc1=h_ls.

[6] AWS's Team, What is Multi-Factor Authentication (MFA)? (2023). URL: https://aws.amazon.com/what-is/mfa/?nc1=h_ls.

[7] Synopsys's Team, Static Application Security Testing (2023). URL: https://www.synopsys.com/glossary/what-is-sast.html
[8] Synopsys's Team, Dynamic Application Security Testing (DAST) (2023). URL: https://www.synopsys.com/glossary/what-is-dast.html
[9] Cloudflare's Team, Why use TLS 1.3? (2023). URL: https://www.cloudflare.com/learning/ssl/why-use-tls-1.3/#:~:text=TLS%201.3%20is%20the%20latest,TLS%20handshakes%2C%20among%20other%20improvements
[10] Sectigo's Team, SHA 256 Algorithm Explained by a Cyber Security Consultant (2023). URL: https://sectigostore.com/blog/sha-256-algorithm-explained-by-a-cyber-security-consultant/
[11] Auth0's Team, Adding Salt to Hashing: A Better Way to Store Passwords (2023). URL: https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/
[12] Imperva's Team, Role-Based Access Control (RBAC) (2023). URL: https://www.imperva.com/learn/data-security/role-based-access-control-rbac/
[13] ProxyDefense's Team, What Is Cryptographic Strength: Definition, Examples & More (2023). URL: https://proxydefense.com/cryptographic-strength/