PROTECTION OF USER CREDENTIALS IN WEB APPLICATION

Stefan Kartunov, PhD
Faculty of Mechanical and Precision Engineering, Technical University of Gabrovo, Bulgaria

Petar Milić, PhD[1]
Faculty of Technical Sciences, University in Priština, Kosovska Mitrovica, Serbia

Kristijan Kuk, PhD[2]
University of Criminal Investigation and Police Studies, Belgrade, Serbia

Uroš Dinić
General Police Directorate of the Ministry of the Interior of the Republic of Serbia

Abstract: Protecting user data in web applications is a challenge that developers face with. Enabling secure user login and overcome of vulnerabilities without exposing sensitive data to attackers is based on the application of modern methods and techniques for protection. This paper first reviews the current situation in this area with an emphasis on important security features. Afterwards, the available systems are evaluated and the obtained results are quantified, from where their advantages and disadvantages can be seen. Another goal of the paper is a deeper insight into the field of user data security. Special attention is given to the importance of password protection for access in web applications.

Keywords: web application security, vulnerabilities, data security, user data protection.

INTRODUCTION

Web applications are key enablers for convenient and on-demand network access resources deployed for various needs. Consequently, any web application is exposed to the attacks that come from the network, thus raising issues with confidentiality, authentication, integrity and security. Vulnerabilities in the web applications are exploited by attackers in order to gain access to the system with the aim to cause loss and harm. Keeping this in mind, the first step in protection of the web applications must be provision of secure login to the system (Ristić, Jevremović, & Veinović, 2013). Secure login represents increasingly significant security issue in the modern web applications that deal with protection of user passwords gathered from database. The most common attack to reveal user password is known as SQLi (SQL injection) and XSS (Cross-Site Scripting). SQLi attack take advantage of unchecked input

1 petar.milic@pr·ac.rs
2 kristijan.kuk@kpu.edu.rs

fields in the web application, in most cases login form, to maliciously tweak the SQL query sent to the backend database (Fonseca, Seixas, Vieira, & Madeira, 2014). Similarly, XSS attack utilize executable code like JavaScript into the user browser, in order to leak user secrets for carrying out operations with the user's privilege on behalf of the adversary. A typical web application which is implemented in PHP, is depicted on the Figure 1. It is assumed that a client interacts with the web application by sending HTTP requests and receiving HTTP responses. In relation to the attacks, SQLi can be related to the HTTP requests while XSS can be related to the HTTP responses.

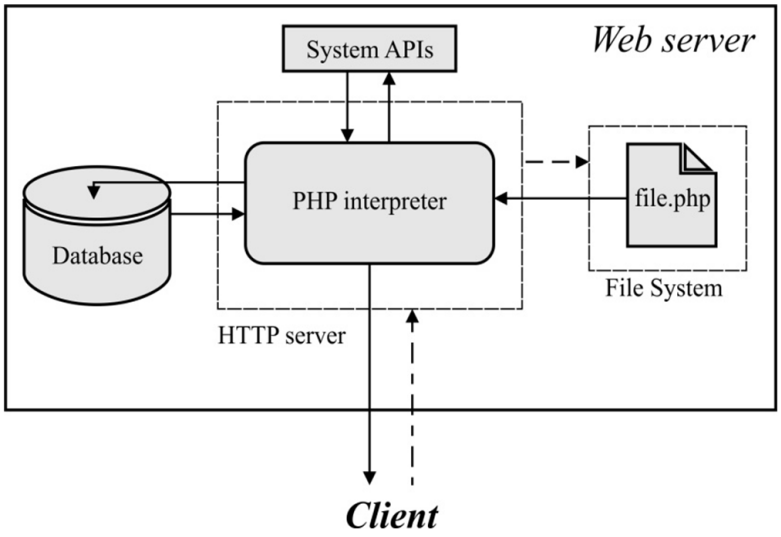

Figure 1. *Typical web application architecture*

Contemporary web applications, such as LinkedIn, Twitter and Sony show that compromising back-end databases can enable attacker access to the confident information (Hunt, 2020). Even such large systems can have security vulnerabilities that can compromise user data. In most cases, user data are compromised due to the improper handle of the variables which are used for gathering of the user entered data. For example, the attacker can exploit variables that supposedly should not be strings (e.g., numbers, dates) for the SQLi attack. Furthermore, revealing of user password can be done by user login in from untrusted machines, such as a friend's computer, the public access machines and others which do not use HTTPS for securing the communication between the client and server.

In this paper we will analyze methods and techniques for secure login in into the web application. By manual literature review, we have identified that protection of user credentials is one of the key enabler for the safe usage of web application. Nevertheless, we will show that current mechanisms of protection is not enough for provision of the desirable level of security. Throughout the paper, the comparative analysis of the available methods and techniques used for securing of the user credential is provided, along with the pros and cons and reference to the OWASP Top Ten as a standard awareness document in the field of the web security. Moreover, we will provide case study of password protection methods and techniques and discuss obtained results. Finally, we conclude the paper with remarks on the future directions of research and give necessary recommendations for the protection of the user credentials.

RELATED WORK

As we mentioned earlier, SQLi and XSS attacks represent the most prevalent security incidents related to the web applications which comes due to the fact that valuable data passes through the web application that uses database (Milić, Kuk, Civelek, Popović, & Kartunov, 2016). If a user input is not handled properly, attackers can insert malicious code which can grant access to the sensitive information and, moreover, cause loss of data. Fonseca & Vieira (2008) came to the conclusion that these two types of attacks can be produced by software faults in development phase of the web applications.

An architecture called *Session Juggler* proposed by Bursztein, Soman, Boneh, & Mitchell (2012) secure user credentials without even entering their long term credentials on the browser. In *Session Juggler* a user first goes to the desired web application, where resulting URL is transferred to the mobile phone where they are asked to login to the site from the phone. By using this approach, *Session Juggler* prevents malware and attackers from hijacking the session by a network sniffer after login. They have shown that on logout, a proper invalidation of session tokens must occur on client and server side. In comparison to the other available tools, *Session Juggler* does not require modification on client side or server side in order to implement the secure login mechanism. For example, research from (Oprea, Balfanz, Durfee, & Smetters, 2004) and (Sharp, Scott, & Beresford, 2006) introduces usage of third parties in the process of protection of user login, consequently requiring modification of client side operations.

Similarly, (Bonneau, Herley, Van Oorschot, & Stajano, 2015) and (Wang, Jian, Huang, & Wang, 2017) propose generation a list of passwords that are only usable one time. This approach requires a server-side modification forcing users to keep a list of passwords for application they want to use. The main problem with this approach is that it does not address the issue that attacker can prevent user from logging out. Considering the usability, deployability and security challenges in the web application authentication will prevent the production of a long list of mutually incompatible password requirements for users. Novel approaches for overcoming these issues rely on applying machine learning in process of the user authentication. Namely, users are faced with requests for second factor authentication (such as one-time code over SMS) when the classifier's confidence is low or the system detects suspicious activity on the application. Nevertheless, a large scale of web applications appear to cope with insecure passwords due to the fact that shortcomings can be covered up with technological smarts in the back-end (Weir, Aggarwal, Collins, & Stern, 2010).

Recent study on securing web applications through login authentication, suggests implementation of hashing of passwords stored in the databases. It was shown that combination of password hashing by using a SHA algorithm and usage of one time passwords increase the security of the authentication and user credentials as well (Maliberan, 2019; Blue, Furey, & Condell, 2017; Seta, Wati, & Kusuma, 2019). Organization such as OWASP (Open Web Application Security Project), PCI-SSC (Payment Card Industry – Security Standards Council), IETF (Internet Engineering Task Force) and IEEE (Institute of Electrical and Electronics Engineers) provide developers and security specialists with necessary recommendations for establishing secure authentication mechanisms in databases. Their primary advice is to avoid usage of an outdated hashing algorithms such as MD5, SHA-128 and incorporation of salt value in stored hashed passwords. As there are no perfectly secure system, setting enough barriers and making the process guessing the password more difficult for the attacker, a higher level of security of user credentials can be achieved with a minimum investment and changes in the existing systems. Also, this barriers will overcome the problem of the rainbow table attack where the attackers must generate a huge rainbow table in order to exploit the system (Wen Chai, 2018). Moreover, a distributed hash crack will be significantly slowed down (Zonenberg, 2009).

ANALYSIS OF METHODS AND TECHNIQUES FOR SECURING USER CREDENTIALS

The importance of protection of the user credentials, especially passwords is a challenge for developers and administrators. In order to bring them closer to the pros and cons of the available tools, in this section we will try to define the criteria for analyzing the available methods, techniques and mechanisms for user data protection. Securing the user data in database-driven web applications is necessary in achieving adequate level of security in such system.

A. *One-sided encryption and hashing*

By using this approach, the password whether it is encrypted or hashed, still exists in the network traffic in order to reach the service. Although they are stored in a database, the attackers will have a chance to succeed to retrieve password back in the plain text by brute force attack, if weak algorithms such as MD5 and SHA-128 is used. Furthermore, encryption methods such as DES, RC4 and Blowfish is proven as weak during to their nature of weak keys used for ciphering. With the progress of the technology, sophisticated attacks utilize GPU chips for brute force attacks as well as the possibility of leasing resources where such system becomes extremely insecure (Zonenberg, 2009).

B. *Two-sided encryption*

Extending the approach of one-sided encryption and hashing with addition of new feature increases the level of security of user credentials. This extensions is reflected in the introduction of one time passwords (OTP). They are valid for only one login session or transaction (Wen Chai, 2018). The advantage of this system is that the algorithm is different of each user's token preventing in that way breaking the algorithm by attackers. Similarly, by using two factor authentication organization can easily increase the level of security of the system with minimal changes in existing authentication system. In this way, attackers are faced with a more difficult step to access a target (Wang, He, Wang, & Chu, 2014).

C. *Security policies*

Anis (2018) has proposed usage of a security policies which work toward preventing attack on web applications and consequently preventing revealing of the user credentials. The idea behind this approach is to ascertain that no user inputs can be received or sent back without going through escaping and encoding mechanisms. Also, OWASP (2020) suggests the use of policy called 'Principle of Least Privilege' which checks all the resources used by the web application against integrity preventing in that manner their alterations. Moreover, OWASP claims that SQLi and XSS attack can be prevented with 'Input Sanitization' and 'Output Validation' policies. Keeping in mind that user credentials can be used for access to multiple services via single sign on, protection of credentials in that case becomes a challenge during to the heterogeneity of the environment that is responsible for secure interoperation between different parties involved (Camath, Liscano, & El Saddik, 2006).

D. *Segmented protection*

Securing a user credentials with segmented protection is based on the encryption of the password with strong hashing algorithms and wrapping it with randomly generated salt values (Ristić, Jevremović, & Veinović, 2013). Random generation of salt values, can be achieved by using PRNG (Pseudo

Random Number Generator) and TRNG (True Random Number Generator) (Adamović, Milenković, Šarac, & Radovanović, 2010). After that process, the final value is divided into two parts and stored into two different databases. It was proven that this approach can improve security, aid compliance, preserve privacy and protect users whose information is stored on insecure system (Blue, Furey, & Condell, 2017). Furthermore, SQLi attack can be mitigated via this approach.

CASE STUDY – PASSWORD PROTECTION BY HASHING FUNCTION

For the purposes of evaluation of password protection by using hashing functions, we have chosen to use phpMyAdmin, a popular database administration tool. This tool implements password storage mechanisms along with a usage of different password hashing functions such as SHA-256, MD5 and BCrypt. Furthermore, two most commonly used techniques for breaking password on these tool is *dictionary attack* and *brute force attack*, which are also used in our experiments. We have conducted experiments with Kali Linux dictionary named "*rockyou.txt*" which is a built-in wordlist with the aim to help any stakeholder to perform different types of password cracking attacks. Moreover, if password is not cracked with dictionary attack method, we try brute force attack with Kraken, a free Windows recovery tool. In Table 1 shows the results of the performed experiments.

Table 1. *Password cracking results*

Method	Password	Hashing function	Time	Can be broken
Dictionary	"kraken"	MD5	578s	yes
Brute force	"fun"	MD5	2692s	yes
Dictionary	"college"	SHA256	1033s	yes
Brute force	"dog"	SHA256	5299s	yes
Dictionary	"batman"	SHA256 with salt	-	no
Brute force	"kraken"	SHA256 with salt	-	no
Dictionary	"password"	BCrypt	847s	yes
Brute force	"dog"	BCrypt	4446s	yes
Dictionary	"hello"	BCrypt with salt	-	no
Brute force	"fun"	BCrypt with salt	-	no

As can be noticed from the Table 1, old types of hashing functions such as MD5 and SHA-256 are vulnerable to the both attack methods, which is in line with our discussion in previous section of the paper. Whatever method is used, it is a matter of time before the password is revealed. Also, the brute force attack takes more time than the dictionary attack for password cracking. It is interesting to mention that usage of the BCrypt hashing function with 4 iterations, extends the time needed for the brute force attack to reveal a password at one hour. In today's world of modern computers that have better and better performance, the time required to damage the system by applying this technique is no longer so significant (Figure 2.). One of the reasons for such situation is utilization of GPU's.

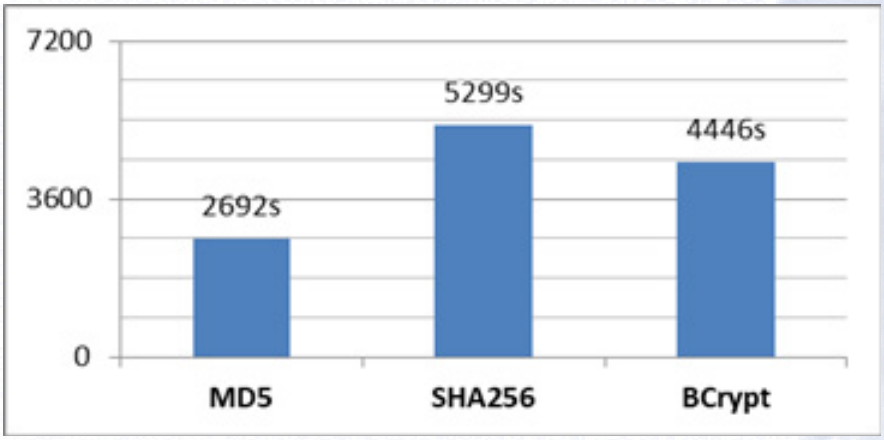

Figure 2. *Time to brute force the password "dog" hashed by different hashing functions on a Nvidia GeForce GTX 1050 3 GB GPU*

Regardless of the hash function applied, by applying the "salt" technique, the password can be made resistant to both types of attacks. Our experiments confirmed this assumption. Addition of the salt values to the password and application of hashing functions such as SHA-256 and BCrypt, puts in front of the attacker a difficult task. Following OWASP Guidelines (OWASP, 2020) for proper implementation of credential-specific salt values, a unique salt must be created upon creation of each credential.

CONCLUSION

In this paper we have analyzed current trends in protection of user credentials in web applications. Also we have provided evaluation of novel technique for protection of user credentials with usage of salt values in hashing process. Utilization of salts in hash functions increases a level of security of user passwords. By following that approach they become unique and attackers cannot reverse them without a lot of effort. Keeping in mind that a number of iterations are performed inside the each hash function on a clear text password, increasing the number of iterations further increases password security, making the attacker's job more difficult, requiring a large amount of computational power and time. Obtained results shows that attacker is faced with a "bottleneck" in the process of password cracking.

For the purpose of minimization of the risk and prevention of attack, SANS Institute (SANS Insitute, 2020) recommends not to use any SHA hashing mechanism under 512 for password hashing due to the reason that they are fast hashes. Application of salt values, make a hash function to look non-deterministic which, as we said early, protects user credentials stored in database.

Further research in this area may lead to the identification of novel approaches and techniques that can increase password stored in databases which are used for accessing web applications. For example, a broader study on evaluation of different hashing and encryption methods with combination of multiple perturbation and application of salt values can identify potential new directions for protection of user credentials.

REFERENCES

1. Adamović, S., Milenković, M., Šarac, M., & Radovanović, D. (2010). Generators of random sequences and their impact on security. *In Proceedings of the INFOTEH 2010* (pp. 820 - 822). Jahorina, BiH: IEEE.
2. Anis, A. (2018). *Securing web applications with secure coding practicies and integrity verification.* Ontario, Canada: Queen's University.
3. Blue, J., Furey, E., & Condell, J. (2017). A Novel Approach for Secure Identity Authentication in Legacy Database Systems. *In Proceedings of the XXVIII Irish Signals and Systems Conference* (pp. 1-6). Killarney, Ireland: IEEE.
4. Bonneau, J., Herley, C., Van Oorschot, P. C., & Stajano, F. (2015). Passwords and the Evolution of Imperfect Authentication. *Communications of the ACM, 58*(7), 78 - 87.
5. Bursztein, E., Soman, C., Boneh, D., & Mitchell, J. C. (2012). SessionJuggler: Secure Web Login From an Untrusted Terminal Using Session Hijacking. *In Proceedings of the XXI International Conference on World Wide Web* (pp. 321 - 330). Lyon, France: ACM.
6. Camath, A., Liscano, R., & El Saddik, A. (2006). User-Credential Based Role Mapping in Multi-domain Environment. *In Proceedings of the International Conference on Privacy, Security and Trust* (pp. 62 - 69). New York, USA: ACM.
7. *CVE*. (2017). Retrieved 2017, from Common Vulnerabilities and Exposures: https://cve.mitre.org/
8. *CVSS*. (2017). Retrieved 2017, from Common Vulnerability Scoring System: https://www.first.org/cvss
9. *CWE*. (2017). Retrieved 2017, from Common Weakness Enumeration: https://cwe.mitre.org/
10. Fonseca, J., & Vieira, M. (2008). Mapping software faults with web security vulnerabilities. *In Proceedings of the IEEE International Conference on Dependable Systems and Networks* (pp. 257-266). Anchorage, Alaska, USA: IEEE.
11. Fonseca, J., Seixas, N., Vieira, M., & Madeira, H. (2014). Analysis of Field Data on Web Security Vulnerabilities. *IEEE Transactions on dependable and secure computing, 11*(2), 89 - 100.
12. Hunt, T. (2020, July). Retrieved from https://www.troyhunt.com/brief-sony-password-analysis/
13. Maliberan, E. V. (2019). Modified SHA1: A Hashing Solution to Secure Web Applications through Login Authentication. *International Journal of Communication Networks and Information Security, 11*(1), 36-41.
14. Milić, P., Kuk, K., Civelek, T., Popović, B., & Kartunov, S. (2016). The Importance of Secure Access to E-government Services. *In Proceedings of the International Conference "Archibald Reiss Days"* (pp. 307-3016). Belgrade, Serbia: Academy of Criminalistic and Police Studies.
15. Oprea, A., Balfanz, D., Durfee, G., & Smetters, D. K. (2004). Securing a Remote Terminal Application with a Mobile Trusted Device. *In Proceedings of the XX Annual Computer Security Applications Conference* (pp. 438 - 447). Tucson, USA: IEEE.
16. OWASP. (2020). *Security by Design Principles - OWASP*. Retrieved 06 30, 2020, from https://www.owasp.org/index.php/Security_by_Design_Principles#Principle_of_Least_privilege

17. Ristić, N., Jevremović, A., & Veinović, M. (2013). System of segment protection of user data in the web applications. *In Proceedings of the XII International Conference "INFOTEH". 12*, pp. 915 - 918. Jahorina, BiH: IEEE.

18. Seta, H., Wati, T., & Kusuma, I. C. (2019). Implement Time Based One Time Password and Secure Hash Algorithm 1 for Security of Website Login Authentication. *In Proceedings of the International Conference on Informatics, Multimedia, Cyber and Information System* (pp. 115 - 120). Jakarta, Indonesia: IEEE.

19. Sharp, R., Scott, J., & Beresford, A. R. (2006). Secure mobile computing via public terminals. *In Proceedings of the International Conference on Pervasive Computing* (pp. 238 - 253). Berlin, Germany: Springer.

20. Wang, D., He, D., Wang, P., & Chu, C.-H. (2014). Anonymous Two-Factor Authentication in DistributedSystems: Certain Goals Are Beyond Attainment. *IEEE Transactions on Dependable and Secure Computing, 12*(4), 428 - 442.

21. Wang, D., Jian, G., Huang, X., & Wang, P. (2017). Zipf's Law in Passwords. *IEEE Transactions on Information Forensics and Security, 12*(11), 2776 - 2791.

22. Weir, M., Aggarwal, S., Collins, M., & Stern, H. (2010). Testing Metrics for Password Creation Policies by Attacking Large Sets of Revealed Passwords. *In Proceeding of the XVII ACM Conference on Computer and Communication Security* (pp. 162-175). Chicago, USA: ACM.

23. Wen Chai, C. (2018). *Secure login authentication system.* Universiti Tunku Abdul Rahman.

24. Zonenberg, A. (2009). *Distributed Hash Cracker: A Cross-Platform GPU-Accelerated Password Recovery System.* New York, USA: Rensselaer Polytechnic Institute.