Name: Ayangade Adeoluwa Joseph.
Student ID: w22038122.
Course – KF7013.
Module Tutor – Emma Anderson.

INTRODUCTION

As a result of growing web security threats, the safety and security of web applications are crucial concerns that all web owners and developers share. Confidential information of businesses and their customers can be exposed via little bugs in the code, as malicious individuals are constantly looking for new ways to steal data.

To understand the security measures put in place to address vulnerabilities in our web application, we need to identify the software stack the web application uses and access points that a potential hacker could seek to exploit through them. The software stack used by the business is Hyper Text Markup Language (HTML), Cascading Style Sheets (CSS), PHP, and a MySQL database. Potential access points that a malicious individual could take advantage of are the HTML Forms either for booking a trip or for registering with the business, the database via SQL queries, and the execution of malicious PHP code on the server. It is important to prevent these access points from common web attacks like SQL Injections, Cryptographic failures, Identification, and Authentication failures, and so on (OWASP, no date).

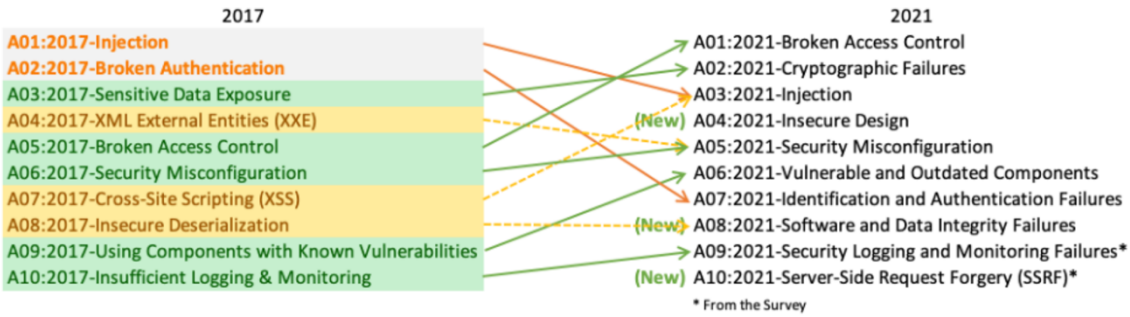

Figure 1.1: OWASP Top ten application security (OWASP, no date).

Here, I discuss what security measures were put in place to guard against these cyber threats for my web application and make recommendations on how the web application can be further secured from them.

ANALYSIS

Data Validation and Sanitization

The website has multiple forms that are used for user authentication, registration, and booking an excursion. It is important to validate the form data supplied by the user and make sure the data provided meets our criteria. For example, emails and passwords must follow a specific format. Also, it is a general rule of thumb for the server not to trust anything sent by the client, this includes form data supplied by the user or query parameters used while navigating between pages, hence, sanitization must be done to remove potentially dangerous characters or symbols from the data before being persisted or used in an SQL query (Balzarotti et al., 2008). PHP provides helper functions for both validation and sanitization, the *htmlSpecialChars,* and *filter_var* functions (PHP, no date).

Prepared Statements

SQL injection is a vulnerability that enables hackers to modify database queries that a web application makes, it usually results in retrieving business or customer data that they would not typically be able to access by tricking the web application into making malicious queries. This threat can be mitigated by using prepared statements. A prepared statement is an SQL query with the variables replaced by a placeholder or parameter (“?”), the placeholder is then replaced by the variable before the query is executed, since the SQL query is explicitly set before execution, it cannot be modified (Amirtahmasebi et al., 2009). PHP provides helper

functions like *mysqli_prepare*, *mysqli_stmt_bind_params,* and *mysqli_stmt_execute*, *which* can be used to write prepared statements and execute them (PHP, no date).

Hashing

Sensitive customer data like passwords should not be stored plainly in databases as those databases can be breached or have their records fetched via a SQL injection attack, hence why the data should be hashed (Hatzivasilis, 2017). Hashing is a form of one-way encryption that involves the transformation of any string into a random set of characters. The hashed data will only be used for verification purposes for example during user authentication. PHP provides helper functions like *password_hash* and *password_verify*, to hash customer passwords before being persisted in the database ((PHP, no date).

Session Management and Protections

Our Web application relies on creating sessions when a user is logged in, this allows us to securely handle multiple requests made by the user to our website and allow permission to restricted web application pages. If session information is not properly configured and managed, attackers can compromise the session tokens and assume the identities of user accounts. Each created session variable must be deleted when no longer needed or when the user is logged out of the application (Vlsaggio and Blasio, 2010). PHP provides helper functions like *session_unset* and *session_destroy* to achieve this ((PHP, no date).

Restricted Access

Access to various parts of the web application must be restricted to only authorized users of the website. Hence, various levels of permissions should be granted to visitors of the website. Users should have full access to the customer-facing web pages, and non-users should not be

allowed to access sensitive customer-facing web pages like the dashboard or excursion booking pages.

RECOMMENDATIONS

Database Encryption

Databases can be regarded as large reservoirs of data; this makes them a prime target of most cyber-attacks. It is recommended that our web application should consider encrypting only important data stored in the database using the Advanced Encryption Standard (AES-128) to ensure that in the event of a breach, the data is unusable to hackers (Mattsson, 2005). AES-128 is a form of symmetric block cryptographic cipher that encrypts and decrypts data using chunks of bytes (128 bits) (Stallings, 2002). It is fast and almost impossible to crack compared to alternatives (Mattsson, 2005).

Environment Variables

The configuration variables used within the application for connecting to the database are stored as plain strings within the source code. This is considered bad practice because if ever the company codebase is leaked, those sensitive database credentials would be exposed to malicious hackers. It is recommended environment variables are used, and configuration values are stored elsewhere on the server, for example, at the server's topmost root directory. This will ensure the web application secrets are not exposed if the source code ever gets leaked (Kinsta, 2022).

HTTPS (Hyper Text Transfer Protocol Secure) Internet Protocol

Currently, the web application uses the Hyper Text Transfer Protocol (HTTP) to share data over the internet, because of this, communication between the business servers and the clients

is sent unencrypted and unverified over the network. Malicious individuals can easily intercept the HTTP request and responses and steal business or customer data (Callegati et al., 2009). It is recommended the business makes use of the Hyper Text Transfer Protocol Secure (HTTPS) instead. HTTPS uses the Secure Sockets Layer encryption to encrypt and digitally sign data shared over the network. It is far more secure than using HTTP (Cloudflare, no date).

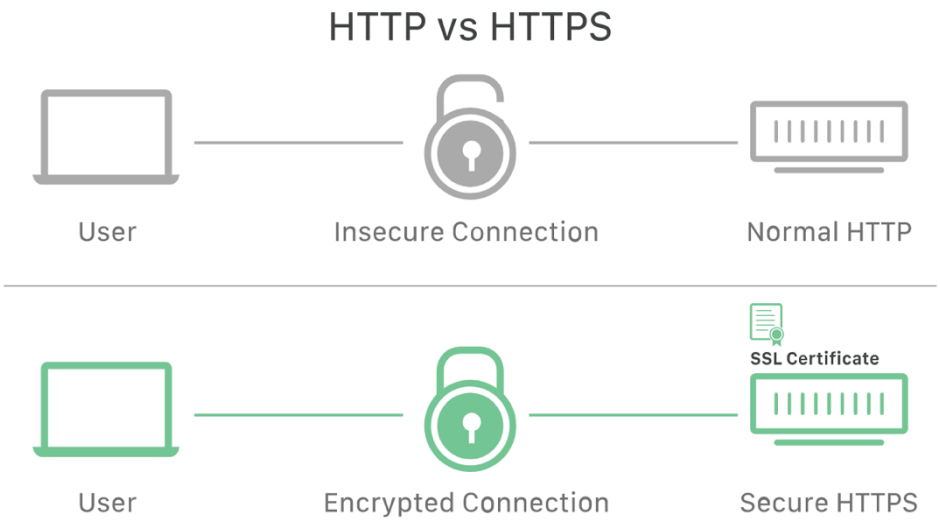

Figure 1.2: Difference between HTTP and HTTPS protocols (Cloudflare, no date)

REFERENCES

OWASP (no date) *Owasp Top Ten*, *OWASP Top Ten | OWASP Foundation*. Available at: https://owasp.org/www-project-top-ten/ (Accessed: January 3, 2023).

Balzarotti, D. *et al.* (2008) "Saner: Composing static and dynamic analysis to validate sanitization in web applications," *2008 IEEE Symposium on Security and Privacy (sp 2008)* [Preprint]. Available at: https://doi.org/10.1109/sp.2008.22.

PHP (no date) *Hypertext preprocessor*, *php*. Available at: https://www.php.net/ (Accessed: January 3, 2023).

Hatzivasilis, G. (2017) "Password-hashing status," *Cryptography*, 1(2), p. 10. Available at: https://doi.org/10.3390/cryptography1020010.

Amirtahmasebi, K., Jalalinia, S.R. and Khadem, S. (2009) "A survey of SQL Injection Defense Mechanisms," *2009 International Conference for Internet Technology and Secured Transactions, (ICITST)* [Preprint]. Available at: https://doi.org/10.1109/icitst.2009.5402604.

Vlsaggio, C.A., and Blasio, L.C. (2010) "Session management vulnerabilities in today's web," *IEEE Security & Privacy*, 8(5), pp. 48–56. Available at: https://doi.org/https://doi.org/10.1109/MSP.2010.114.

Stallings, W. (2002) "The Advanced Encryption Standard," *Cryptologia*, 26(3), pp. 165–188. Available at: https://doi.org/10.1080/0161-110291890876.

Mattsson, U.T. (2005) "Database encryption - how to balance security with performance," *SSRN Electronic Journal* [Preprint]. Available at: https://doi.org/10.2139/ssrn.670561.

Callegati, F., Cerroni, W. and Ramilli, M. (2009) "Man-in-the-middle attack to the HTTPS protocol," *IEEE Security & Privacy Magazine*, 7(1), pp. 78–81. Available at: https://doi.org/10.1109/msp.2009.12.

Cloudflare (no date) *Why is HTTP not secure? | HTTP vs. HTTPS | cloudflare*, *Why is HTTP not secure? | HTTP vs. HTTPS*. Available at: https://www.cloudflare.com/learning/ssl/why-is-http-not-secure/ (Accessed: January 3, 2023).

Kinsta (2022) *Environment variables: What they are and how to use them*, *Kinsta®*. Available at: https://kinsta.com/knowledgebase/what-is-an-environment-variable/#:~:text=System%20environment%20variables%20reside%20on,required%20to%20fiddle%20with%20them. (Accessed: January 3, 2023).