Northumbria University NEWCASTLE

Security Report for TravelWise website

Author: Emmanouil Marketos

Student ID: 21050558

Word Count: 956

Professor: Dr Emma Anderson

A report submitted as partial fulfilment of
the requirements for course PE7045 – Secure Web Development

at
Northumbria University
September 2022

1. Introduction

The purpose of the present report is to illustrate and justify the measures implemented in order to ensure the security, to a certain extent, of the TravelWise web application. Having said that, the web security topic is huge and can have very sophisticated techniques in order to mitigate the negative effects of security breaches concerning authentication and access control, SQL injection, cross-site scripting (XSS), information leakage and cross-site request forgery (Stuttard and Pinto, 2011). The following section details the methods applied in the TravelWise web application, to achieve the basic security that at the very least all websites should have.

2. Mechanisms for user access handling

2.1 Access Control: Unauthorized users are not permitted to visit certain pages according to the assignment specifications, they first need to authenticate themselves via the login process and then they are granted full access to the website. Access control is achieved by ensuring the appropriate session value ('logged-in') is stored in the generated session file.

2.2 Authentication: Users need first to register on the website in order to take advantage of its functionalities. A series of data validation checks for each registration form field ensure first that the user's login details have the minimum amount of complexity, for example, minimum string length and also complexity patterns defined through regular expression matches. Password confirmation check, as well as password hashing, is used to store the password in the database in a secure way that will prevent the password from being disclosed in an event of a database leak.

2.3 Sessions: After the successful login, the next security measure is to identify the HTTP requests of each user to the server and distinguish their interactions with the TravelWise web application (Stuttard and Pinto, 2011 p.19). This is being achieved with the use of sessions which for the purposes of the TravelWise website store a variable showing when the user is logged in as well as their unique username. Upon logout the session is unset and destroyed, the user is then redirected to the login screen in order for new authentication.

3. SQL injection and XSS defence

As mentioned above the user is prompted at various occasions (registration, login, search bar) to insert data into forms which will then be used to either retrieve or insert data into the database. This type of user interaction involves a serious risk of SQL injection attacks that can cause disclosure of sensitive information, data modification even permanent data loss (SQL Injection | OWASP Foundation, 2022). To prevent this, special characters and single quotes are either removed or encoded and also user input is never used directly, it is always being bound and parametrized through PHP functions for mySQL using prepared statements for safely inserting the user input into the database. Similarly for stored XSS type vulnerabilities appropriate filter, to sanitize the user input from any HTML tag that can be used to embed malicious JavaScript code is utilized on all data provided by users. Moreover, POST rather than GET method is used to retrieve the user input from the super global arrays $_POST and $_GET respectively as the latter is considered less safe for the reason that data from the user request to the HTTP server are displayed in the URL (HTTP Methods GET vs POST, 2022). Hence, mitigating the risk of DOM and reflected XSS attacks (Stuttard and Pinto, 2011 p.448). Additionally, on one occasion where the URL from bookAccoForm.php had to be directly used to identify the chosen accommodation ID, the PHP filter for sanitizing the URL was applied. Finally, after the implementation of the security measures described, Black-Box testing was carried out in order to make sure that the defensive mechanisms were accomplishing their task. This involved user inputs that could have a malicious outcome for example single quote as well as <script> or other kinds of HTML tag inputs.

4. Security Improvements

As mentioned previously, only the basics of web security measures have been implemented. That said there is, understandably, considerable room for improvement. To start with, the source code needs to be reviewed and enhanced in terms of security and validation checks (White-Box testing approach). For all web applications storing customer personal data and potentially payment details, firewall protection must be the very first line of defence especially when URL injection attacks might take place so that dangerous network traffic can be filtered out. According to Nuno and Serrao (2011), one of the most common vulnerabilities is insufficient authentication and session management. Multiple stage sign-in process as well as additional credentials and authentication on restricted pages that contain user personal information. Another improvement in the security of the website would be to implement validation on the server side too. As it stands, TravelWise web application has only validation on the client side, as stated by Fanxing Kong (2016) and Stuttard, D. and Pinto (2011) a way to further secure the client-server exchange of messages is to use the SOAP protocol to also sanitize output to the client from the server.

5. Conclusion

With the continuously and rapidly growing development of web application platforms, especially after the pandemic, from social media and online shopping to digital banking and public services, the need for securing sensitive client information and personal details is a necessity. With this amount of important private information stored on the web applications servers, it is expected that data leaks and cyber-attacks will only get more severe and larger in scope. Thus, it is vital for all web application providers to increase their efforts for securing their platforms. Secure code training for their employees, clear security requirements, as well as post-deployment security assessments, will have to be some of the basic initiatives to keep the security standards high.

6. References

[1] Stuttard, D. and Pinto, M., 2011. The Web Application Hacker's Handbook, 2nd Edition.

[2] Owasp.org. 2022. SQL Injection | OWASP Foundation. [online] Available at: <https://owasp.org/www-community/attacks/SQL_Injection> [Accessed 21 September 2022].

[3] W3schools.com. 2022. HTTP Methods GET vs POST. [online] Available at: <https://www.w3schools.com/tags/ref_httpmethods.asp> [Accessed 22 September 2022].

[4] Teodoro, Nuno & Serrão, Carlos. (2011). Web application security: Improving critical web-based applications quality through in-depth security analysis.

[5] S. Kumar, R. Mahajan, N. Kumar and S. K. Khatri, 2017. A study on web application security and detecting security vulnerabilities

[6] Fanxing Kong, 2016. Research on Security Technology based on WEB Application