

Testing of Data Security Quality on Web Based Applications

School of Applied Computing, Computer Networking and Cybersecurity, University of Wales Trinity Saint David (Swansea Campus/IQ Building

Kian Harper-Evans (2204995@student.uwtsd.ac.uk) (Kianmharper10@outlook.com)

Introduction:

This project intends to address the large number of cases related to data security breaches of numerous WBA’s leading to the exposition of sensitive user data and sensitive data of organizations leading to the exploitation of user data for nefarious purposes and the loss of profits/work progress of many organizations. This will be done through security testing of sample online shopping WBA. The results from such tests will lead to this projects evaluation and the formation of a detailed updated security architecture and plan with the purpose of reducing the likelihood of security breaches in future.

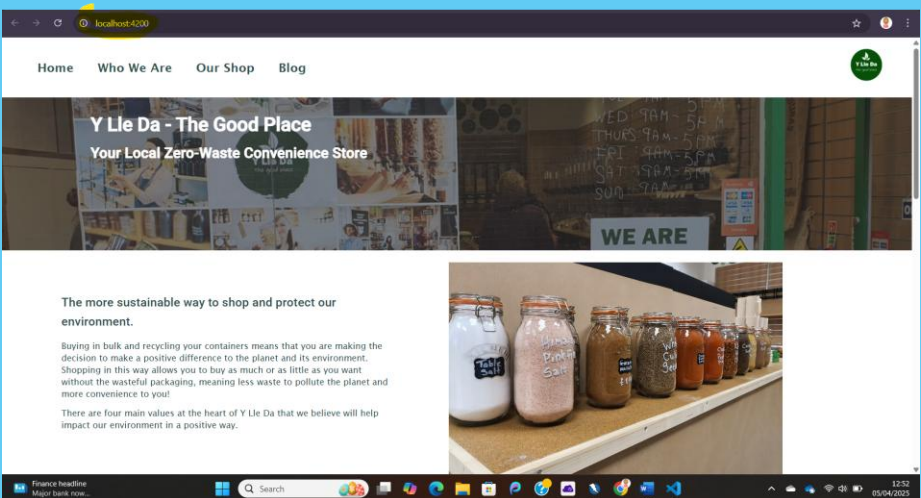
Tools Used in Testing Process:

- WFUZZ – WSL (Windows Subsystem for Linux) based Fuzzing Tool used to discover hidden files within applications
- OWASP ZAP – Automated DAST (Dynamic Application Security Testing) Penetration Testing Tools for general Vulnerability Analyses.
- Vulnerability Scanner – White-box based vulnerability scanner used for analysing application source Code
- Black Widow – (WSL) based crawling tool used to find vulnerabilities possibly missed by ZAP

Tools Yielded Successful Results:

- OWASP ZAP – YES
- WFUZZ – YES
- Vulnerability Scanner – NO
- Black Widow – NO

Note: ALL WSL TOOLS RUN ON UBUNTU TERMINAL



Tested applications shown above

ZAP Test Results:

Left – Ylldela APP – local host
Right– GetFit.wales – live host

Total Vulnerabilities: 43		
Vulnerability	Severity	TTD (Time to Detect)
Failure to define directive with no fallback (3)	Medium	>1 minutes (Near instant auto scan)
CSP Header not set	Medium	>1 minutes (Near instant auto scan)
Cross-domain misconfiguration (12)	Medium	>1 minutes (Near instant auto scan)
Missing anti-clicking header	Medium	>1 minutes (Near instant auto scan)
Server leaks information (12)	Low	>1 minutes (Near instant auto scan)
X-content type options header missing (9)	Low	>1 minutes (Near instant auto scan)
Suspicious comments (4)	Informational	>1 minutes (Near instant auto scan)
Modern web application alert	informational	>1 minutes (Near instant auto scan)

Vulnerability	Severity	TTD (Time to Detect)
CSP Failure to define Directive with no fallback (3)*	Medium	>1 minutes (Near instant auto scan)
Content Security Policy (CSP) header not set*	Medium	>1 minutes (Near instant auto scan)
Cross-domain misconfiguration (12)*	Medium	>1 minutes (Near instant auto scan)
Hidden File Found (4)	Medium	>1 minutes (Near instant auto scan)
Missing Anticlickjacking Header (4)*	Medium	>1 minutes (Near instant auto scan)
Cross-Domain Javascript source file inclusion (3)	low	>1 minutes (Near instant auto scan)
Server leaks information via "X-Powered-by" HTTP response header fields (12)	low	>1 minutes (Near instant auto scan)
X-Content-Type-Options Header Missing (9)	low	>1 minutes (Near instant auto scan)
Information Disclosure - Suspicious comments (9)	Informational	>1 minutes (Near instant auto scan)
Modern Web application (4)	Informational	>1 minutes (Near instant auto scan)
Re-Examine Cache-control directives (3)	Informational	>1 minutes (Near instant auto scan)

WFUZZ Results:

```
=====
* Wfuzz 3.1.0 - The Web Fuzzer
=====
Target: https://getfit.wales/fuzz
Total requests: 4766

=====
ID           Response  Lines  Mod    Chars  Payload
=====
0000000007: 200      21 L   1102 W   20615 Ch  "wall-knove/autocconfig/sell"
0000000008: 200      21 L   1102 W   20615 Ch  "wall-knove/AssetLinks.json"
0000000009: 200      21 L   1102 W   20615 Ch  ".bash_history"
0000000010: 200      21 L   1102 W   20615 Ch  ".cshrc"
0000000011: 200      21 L   1102 W   20615 Ch  ".git_release"
0000000012: 200      21 L   1102 W   20615 Ch  ".psql"

Total time: 12.70902
Processed Requests: 4746
Filtered Requests: 9
Requests/sec.: 373.4354
```

Analyses:

The results from this experiment confirm many things. The first of which being that Black Widow and Vulnerability Scanner/all extensions for VS code are unreliable and unfit for use in the securities Framework. Secondly that tools such as WFUZZ and ZAP are the most suitable used in this experiment for finding reliable results. Finally, that several changes are required in the configuration of both applications to defend against data leakage and malicious attacks . Getfit.wales was found to contain publicly available files such as .env .bash history and .directory files which leave its systems compromised. Ylldela app is left vulnerable due to reasons usually related to the CSP (Content Security Policy) Header being misconfigured.

Securities Framework (Recommendations):

From these tests it can be drawn that the security testing framework should include the following tools:

- OWASP ZAP:
ZAP proved to be one of the most reliable and simple to use tools in the lineup of tools in this project, which mirrors its performance documented in the study [19]. It provides a comprehensive breakdown of each vulnerability found and provides a useful severity ranking that will help administrators prioritize resources and time spent fixing issues. It’s the best DAST penetration testing tool option.
- WFUZZ:
WFUZZ was the most successful of the WSL environment tools. It is useful for locating specific hidden files that can leave an application vulnerable, that tools such as ZAP will miss.

Changes/solutions

This section of the report is to look at the results from all the tests for each application and provide the solutions for the vulnerability. This will be done with a bullet point list with each point being a change or solution to consider.

Yedella App:

- Ensure servers in use are properly configured to set content security policy headers.
- Restrict unauthorized access to sensitive data possibly using ‘IP address whitelisting’.
- Configure ‘Access Control’ headers to be more restrictive or remove ‘CORS’ headers.
- Ensure that either ‘content-security-policy’ or ‘X Frame Content’ HTTP headers are set to one or all the web pages.

GetFit.Wales:

- Take into consideration whether components that compose hidden files are essential.
- If hidden files are essential then make sure strict authorization is required to access it, if not disable the feature.
- Make sure that all Javascript source files are sourced from exclusively trusted sources.
- Ensure that Javascript sources cannot be influenced via end point users (Consumers).
- Ensure that all application software and hardware devices are configured to suppress ‘X-Powered By’ headers.
- Restrict public/unauthorized access to files such as ‘. Git directory’, ‘.env’ and ‘.history’ files to reduce the likelihood that malicious actors can discover the mechanisms that make up a given application.

Conclusion:

To conclude, All this knowledge has then been used to design a experiment plan that was used to test two WBA’s, one being live hosted and one being local hosted. Testing both types of application then gave a deeper understanding of the best practices for application testing as comparing the results shows that live testing is the better option. The experiment section of this project also revealed that certain tools such as extensions for VS code are not suitable for professionally testing source code as they are prone to failures that a dedicated tool may not be, this leaves further room for this experiment to be expanded to confirm this theory. The experiment section of this report also leaves questions to be answered relating to the performance of tools such as WFUZZ on local hosted apps and Black Widow on both types of apps, which possibly could’ve performed better under different circumstances or lengthy re-configuration which was unable to be performed within the timeframe of this project. Finally, this experiment resulted in the composition of a securities framework which recommended the two successfully implemented tools and a list of recommendations to rectify the discovered vulnerabilities. However, this framework could of benefited from recommendations related to source coding practices which were absent due to the poor performance of all the used white box testing tools.

Literature Cited in This Poster:

[19] Nguyen Thanh Cong, Le Huy Toan, and Ta Minh, “Tổng quan về phân tích tĩnh và phân tích động trong kiểm tra bảo mật ứng dụng,” vol. 99, p. 1, Nov. 2024, doi: 10.54939/1859-1043.j.mst.99.2024.1-11.