Executive Summary

Introduction

This report concludes the security evaluation of readmyblog.co.uk by making recommendations to improve its security according to GDPR and OWASP standards based on prior empirical analysis, which is integrated with vulnerability scanning and research. This is driven by an adapted industrial methodology.

Methodology

To make recommendations, vulnerabilities are identified according to the 2021 edition of the OWASP Top Ten (OWASP, 2021) and UK GDPR principles (ICO, n.d.b). These vulnerabilities are found and addressed by following the vulnerability measurement methodology presented by PwC (2021) which is a process comprised of two steps:

1. Identify
    a. Select appropriate scanning tools
    b. Configure scanners
    c. Scan
2. Evaluate
    a. Analyse vulnerabilities found
    b. Quantify the severity of the vulnerabilities found
    c. Provide recommendations to protect against the vulnerabilities found, ordered by severity

To quantify a vulnerability's severity, PwC recommends using a vulnerability scoring system. We therefore use the CVSS v3.1 framework because it provides standardised calculations of severity (FIRST, 2021a). PwC also notes the importance of compliance with regulations as part of security governance, thus we also evaluate the website's GDPR compliance and make appropriate recommendations.

We make changes and improvements to this methodology to improve its efficacy. During scanning, we search for disclosed vulnerabilities on the Github repository for the software powering the website, which is NucleusCMS. During evaluation, we map found vulnerabilities to OWASP categories and also integrate our empirical findings, which improves recommendation quality by making common attack surfaces easier to detect (Votipka et al., 2018). Lastly, we map vulnerability severity to recommended timeframes for correcting these vulnerabilities (Seo, 2019).

Finally, we discuss all limitations which apply to the execution of this methodology.

Tool Selection

To select tools, we establish a set of criteria which is used to judge a tool's suitability. We prioritised ease-of-use and therefore researched tools which are well-documented, fast, and produce results that can be mapped to categories in the OWASP Top Ten.

We identified three viable tools: OWASP ZAP, Arachni and Burp Suite Professional. We use all three as they have different scanning algorithms, which allows more vulnerabilities to be discovered (Ghazanfar et al., 2021; Awlarijal et al., 2020). Below, we provide academic justification for the use of each individual tool:

- OWASP ZAP scans quickly, is easy to use, and has a good detection rate for web applications (Sagar et al., 2018; Awlarijal et al., 2020; Amankwah et al., 2020).
- Arachni has high crawler coverage and generates a low number of false positives (Alsaleh et al., 2017; Amankwah et al., 2020).
- Burp Suite Professional has a detection rate comparable to OWASP ZAP, and a much lower rate of false positives compared to ZAP and Arachni (Anhar & Suryanto, 2021).

Scan Results

Figure 1 shows the results of scanning the website using the selected tools. Vulnerabilities detected by investigating the Github repository are shown in Figure 2.

ID	OWASP ZAP	ID	Burp Suite Professional	ID	Arachni
Z1	XSS (Reflected)	B1	CSRF	R1	CSRF
Z2	Application Error Disclosure	B2	Password with autocomplete	R2	Cookie No Secure Flag
Z3	"X-Powered-By" in HTTP Response Header	B3	User agent-dependent response	R3	Password with autocomplete
Z4	Debug Error Messages	B4	Input returned in response (reflected)	R4	Sensitive File Disclosure
Z5	Cookie No HttpOnly Flag	B5	Suspicious input transformation (reflected)	R5	Missing 'X-Frame-Options' Header
Z6	Cookie No Secure Flag	B6	Backup file	R6	Missing HSTS
Z7	Cookie No SameSite Attribute	B7	Email addresses disclosed	R7	Cookie No HttpOnly Flag
Z8	Cross-Domain JavaScript Source File Inclusion	B8	Robots.txt file	R8	Interesting response (No 200 or 404 status code returned)
Z9	Incomplete or No Cache-control Header	B9	Cacheable HTTPS response	R9	Cookie set for parent domain
Z10	CSRF	B10	TLS certificate		
Z11	Directory Browsing	B11	Mixed content		
Z12	Timestamp Disclosure	B12	Hidden HTTP 2		
Z13	Charset Mismatch				
Z14	Suspicious Comments				

Figure 1: Vulnerabilities Discovered Using Scanning Tools

ID	Github Issues
G1	File Upload (Github, 2019a)
G2	XSS (Stored) (Github, 2019b)
G3	HTML Injection (Github, 2018)

Figure 2: Discovered Vulnerabilities Through Repository Auditing

Result Analysis

False Positives

We begin the analysis by removing false positives from the list of vulnerabilities found. We identify and explain the false positives below:

Vulnerability ID	Explanation
Z2, Z4, Z11	The information displayed is found in documentation, not the application itself.
Z14	Nothing sensitive is rendered inline after the suspicious comment.
B6	This is not a backup, but a JavaScript library.
B8	Robots.txt reveals no sensitive information about the site.
Z8	This file is not hosted externally.
R4	Config.php cannot be accessed directly.
R8	The website returned the correct HTTP response codes (404 and 403) for the situations identified.
R9	Even though the *lastVisit* cookie is set, it expires immediately because the Max-Age attribute is set to 0.

Figure 3: Explanation of False Positives

Vulnerability Mapping

To contextualise the results, we map each vulnerability found to its related OWASP Top Ten category as shown in figures 4 and 5. OWASP ZAP automatically provides this mapping (ZAP Dev Team, 2022a), while Burp Suite and Arachni map their results to a CWE number, which MITRE maps to the correct OWASP category (MITRE, 2021).

OWASP Category	**Corresponding Vulnerability IDs**
A01:2021 - Broken Access Control	Z3, Z7, Z10, Z12, B1, B7, R1
A02:2021 - Cryptographic Failures	R6, B11
A03:2021 - Injection	Z1, Z13, B4, B5, G1, G2, G3
A04:2021 - Insecure Design	Z9, B9, B12, R5
A05:2021 - Security Misconfiguration	Z5, Z6, B2, B3, R3, R2, R7
A06:2021 - Vulnerable and Outdated Components	N/A
A07:2021 - Identification and Authentication Failures	B10
A08:2021 - Software and Data Integrity Failures	N/A
A09:2021 - Security Logging and Monitoring Failures	N/A
A10:2021 - Server Side Request Forgery (SSRF)	N/A

Figure 4: Discovered Vulnerabilities Mapped to OWASP Top Ten Categories

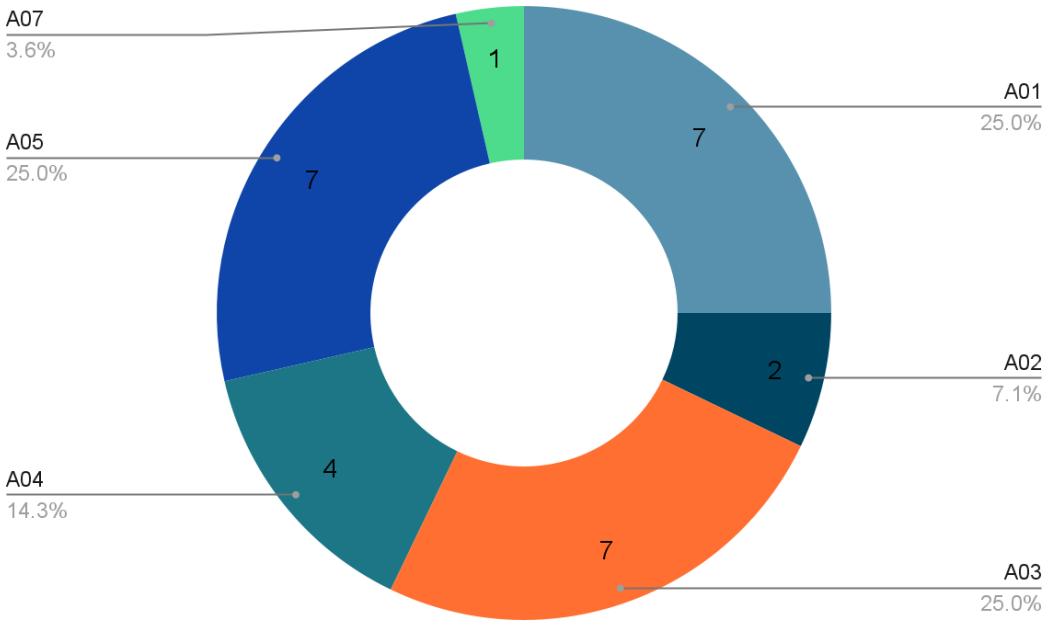

Figure 5: Number of vulnerabilities found per OWASP Category

Scoring

We determine the severity of each vulnerability by using the official CVSS v3.1 calculator (FIRST, 2021b). The calculator uses vulnerability characteristics as input, and then displays a severity score between 1.0 and 10.0. This score is then assigned to a category using the following mapping (FIRST, 2021a):

CVSS Category	CVSS Score
Critical	9.0 - 10.0
High	7.0 - 8.9
Medium	4.0 - 6.9
Low	0.1 - 3.9
None	0.0

Figure 6: CVSS Category Mapping

The following categories were determined for each vulnerability found. Consult the appendix for details on the calculations used.

Vulnerability Severity	Vulnerability ID
Critical	N/A
High	G1
Medium	Z1, Z3, B1, B11, R1, G2, G3, R2
Low	Z5, Z6, Z7, Z9, B2, B3, B9, R3, R5, R6, R7
None	Z10, Z12, Z13, B4, B5, B7, B10, B12

Figure 7: Severity of Found Vulnerabilities

As shown above, some vulnerabilities have no impact and thus have a rating of *None*. This is because they are not vulnerabilities in themselves, but rather are informational, or potential vulnerabilities which require exploitation of other vulnerabilities to be realised (Palanov, 2016).

We now summarise this data by classifying them by severity, across their matching OWASP categories.

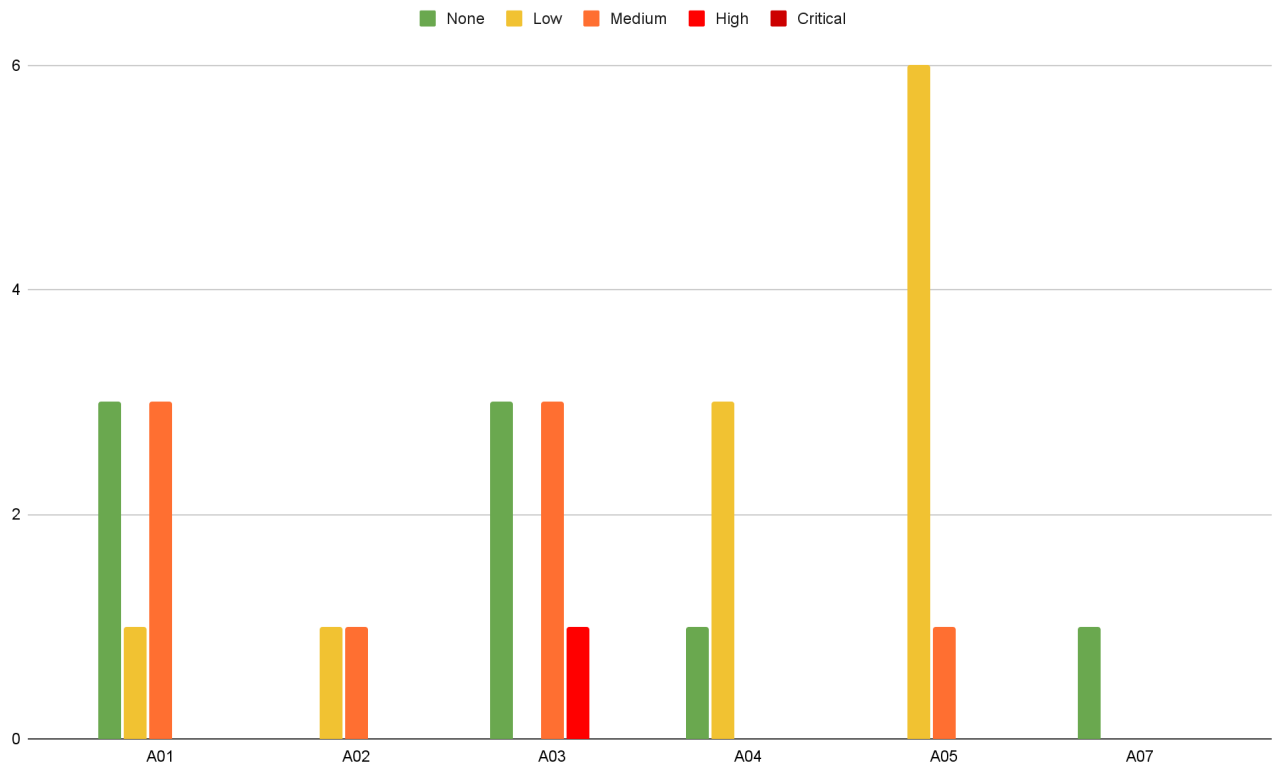

Figure 8: Distribution of Severity Across OWASP Vulnerabilities

Comparison with Empirical Results

We further analyse the data by integrating it with our empirical findings. The table below discusses the empirical vulnerabilities found, and whether the scans confirmed their existence or not.

Empirical Vulnerability ID	Vulnerability Description	OWASP Category	Existence confirmed during scan?
E1	Privilege escalation	A01	No
E2	Unencrypted credential storage	A02	No
E3	Arbitrary code execution through JavaScript written in articles	A03	Confirmed via Github
E4	Improper administrator management	A04	No
E5	Information exposure due to improper configuration	A05	Yes
E6	Usage of an old version of NucleusCMS	A06	Confirmed via Github
E7	Weak password rules	A07	No
E8	Lack of verification of external resources	A08	No (false positive)
E9	Lack of alerts due to suspicious activity	A09	No
E10	Request forgery to gather information	A10	No

Figure 9: Comparison of Empirical and Practical Findings

The majority of differences between the empirical and practical findings are due to a lack of access to privileged areas of the site, as the team was not given login credentials. Vulnerabilities E1, E4 and E7 require access to administrator-only areas to be scanned, whereas to scan for vulnerability E2, access to the underlying web server is required.

During scanning, vulnerability E3 was identified as G3, while two instances of E5 were detected as B3 and Z12. E8 was detected as Z8, although this was a false positive. E9 was not discovered in the scanning phase due to a known limitation (consult the Limitations section). Vulnerabilities E10 and E6 were not discovered and had no problems being scanned for, thus we conclude the website contains neither.

Security Recommendations

From the graphs, it can be seen that OWASP categories A01, A03, and A05 pose the highest threat, however, no universal solutions can be presented for all vulnerabilities in an OWASP category. We therefore present recommendations per vulnerability, ordered by CVSS severity which is mapped to a timeframe recommended to address these issues in (Seo, 2019). Vulnerabilities which have a common recommendation are grouped together.

CVSS Severity	**Timeframe for Remediation**
Critical/High	Short-term
Medium	Mid-term
Low	Long-term

Figure 10: Timeframe Mapping for Remediation

Priority	Vulnerability ID	OWASP Category	Recommendation
High	G1	A03	Do not allow .htaccess files to be uploaded (Gsuhy, 2019)
Medium	Z1	A03	Sanitise arguments given in query URLs.
	Z10, B1, R1	A01	Add anti-CSRF tokens to sensitive forms such as login (Tenable, 2021a).
	B2, R3	A05	Disable autocomplete for the password field of the login form.
	R6	A02	Add the `Strict-Transport-Security` header to HTTP responses (Tenable, 2021c).
	G2	A03	Sanitise the input for the page title parameter when a new article is added.
	G3, E3	A03	Sanitise the input for the body parameter when a new article is added.
Low	Z5, R7	A05	Set HttpOnly to ‘true’ for all cookies.
	Z6, R2	A05	Set the secure flag on the comment cookie to true.
	Z7	A01	Set the SameSite attribute on the comment_email cookie to true.
	Z9, B9	A04	Set no-store, no-cache, and must_revalidate on the HTTP header (ZAP Dev Team, 2022b).
	Z13	A03	Force the XML response to use UTF-8 encoding.
	R5	A04	Add the ‘X-Frame-Options’ header to HTTP responses (Tenable, 2021f).
None	Z3, E5	A01	Do not return any server information in HTTP headers (Tenable, 2019a).
	Z12	A01	Do not display this timestamp.
	B3	A05	Use the same authentication infrastructure across different user agents (PortSwigger, n.d.c).
	B4	A03	Do not render the username that was used after a failed login attempt.
	B5	A03	When “query” is given as an URL parameter, do not render it inline.
	B7	A01	Remove the developer’s email from the webpage.
	B10	A07	Review and update existing TLS configurations.
	B11	A02	Load all external dependencies using HTTPS.

Figure 11: Security Recommendations Ordered by Severity

GDPR Recommendations

To conclude the evaluation, the website's compliance to GDPR is assessed. Currently, the blog does not have a privacy policy, places cookies on a user's computer, and also stores their personal information (name, IP address, and email) when they post a comment. Consequently, it does not comply with the following GDPR principles (ICO, n.d.b):

1. Lawfulness, fairness and transparency
2. Purpose limitation
3. Data minimisation
4. Accuracy
5. Storage limitation
6. Integrity and confidentiality
7. Accountability

To comply with the above principles, the website must first determine what the lawful basis is for storing a user's data (ICO, n.d.a). Thereafter, the website can meet the remainder of the principles by implementing a privacy policy and the corresponding processes. The table below shows which policies need to be established and shown to users in a privacy policy (ICO, n.d.a,; Olsen, 2021; Lubowicka, 2021). Lastly, the website's software must have mechanisms for automated data management based on user requests (Olsen, 2021).

GDPR Principle To Meet	**Policies to Establish, Implement and Display**
Purpose Limitation	What personal information is collected and how it is done Use of cookies, logs, and IP address
Storage Limitation	How long the user's data will be stored
Integrity and Confidentiality	How the data is stored and protected
Individual rights	The data rights available to the user How the user can opt out of data collection How a request for data access can be made
Accuracy	The processes used to guarantee the accuracy of information collected

Figure 12: GDPR Recommendations

Limitations

This report has aimed to comprehensively assess the security of the website, however, limitations were encountered which could be addressed to improve future research:

1. The website uses defence software (Immunify360: Webshield), which had no whitelist for our IP addresses, causing us to be banned from accessing the website after attempted scans.
2. The website owner did not inform us whether or not our scanning tools produced alerts. This is necessary to confirm the existence of OWASP category A09: Insufficient Logging and Monitoring.

However, due to the usage of Immunify360: Webshield, we make the assumption that monitoring is provided.

3. The tools which we selected did not scan for GDPR violations.

Conclusion

Through empirical and practical analysis, the website has been shown to not comply fully with GDPR and OWASP security standards. In this report, we have highlighted the areas of non-compliance, and have provided clear, actionable recommendations that can be implemented to attain compliance and greater security.

References

Alsaleh, M., Alomar, N., Alshreef, M., Alarifi, A. & Al-Salman, A. (2017) Performance-Based Comparative Assessment of Open Source Web Vulnerability Scanners. *Security and Communication Networks 2017:* 1-14. DOI: https://doi.org/10.1155/2017/6158107

Amankwah, R., Chen, J., Kudjo, P. & Towey, D. (2020) An empirical comparison of commercial and open-source web vulnerability scanners. *Software: Practice and Experience* 50(9): 1842– 1857. DOI: https://doi.org/10.1002/spe.2870

Anhar, A. & Suryanto, Y. (2021) 'Evaluation of Web Application Vulnerability Scanner for Modern Web Applications', *2021 International Conference on Artificial Intelligence and Computer Science Technology (ICAICST)*. Yogyakarta, Indonesia, 29-30 June. New Jersey: IEEE. 200-204.

Awlarijal, A., Almaarif, A. & Budiono., A. (2020) 'Vulnerability Assessment for Basic Data of Education Website in Regional Government X - A Black Box Testing Approach', *Proceedings of the 2nd Faculty of Industrial Technology International Congress International Conference.* Bandung, Indonesia, 28-30 January. Indonesia: Faculty of Industrial Technology. 163-168.

FIRST. (2021a) Common Vulnerability Scoring System v3.1: User Guide. Available from: https://www.first.org/cvss/v3.1/user-guide [Accessed 2 February 2022].

FIRST. (2021b) Common Vulnerability Scoring System Version 3.1 Calculator. Available from: https://www.first.org/cvss/calculator/3.1 [Accessed 10 February 2022].

Ghazanfar, I., Abbas, H., Iqbal, W. & Rashid, I. (2021) 'Vulnerability Assessment of Pakistan Government Websites', *2021 International Conference on Communication Technologies (ComTech)*. Rawalpindi, Pakistan, 21-22 September. New Jersey: IEEE. 115-119.

Github. (2018) HTML Injection in Nucleus CMS 3.70. Available from: https://github.com/NucleusCMS/NucleusCMS/issues/84 [Accessed 14 February 2022].

Github. (2019a) File upload vulnerability in Nucleus CMS v3.71. Available from: https://github.com/NucleusCMS/NucleusCMS/issues/96 [Accessed 14 February 2022].

Github. (2019b) Stored XSS - Nucleus CMS v3.71. Available from: https://github.com/NucleusCMS/NucleusCMS/issues/94 [Accessed 14 February 2022].

Gsuhy. (2019) File upload vulnerability. https://shimo.im/docs/Ch9CphJt8XwTvQ3d/read [Accessed 14 February 2022].

ICO. (n.d.a) Lawful Basis for Processing. Available from: https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/lawful-basis-for-processing/ [Accessed 13 February 2022].

ICO. (n.d.b) The principles. Available from: https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/principles/ [Accessed 13 February 2022].

Lubowicka, K. (2021) 10 Elements Every GDPR-Compliant Privacy Policy Should Have. Available from: https://piwik.pro/blog/elements-gdpr-compliant-privacy-policy/ [Accessed 13 February 2022].

MITRE. (n.d.a) CWE-829: Inclusion of Functionality from Untrusted Control Sphere. Available from: https://cwe.mitre.org/data/definitions/829.html [Accessed 14 February 2022].

MITRE. (2021) CWE VIEW: Weaknesses in OWASP Top Ten (2021). Available from: https://cwe.mitre.org/data/definitions/1344.html [Accessed 14 February 2022].

Olsen, N. (2021) The Eight User Rights Under the GDPR. Available from: https://www.privacypolicies.com/blog/gdpr-eight-user-rights/ [Accessed 13 February 2022].

OWASP. (2021) Welcome to the OWASP Top 10 - 2021. Available from: https://owasp.org/Top10 [Accessed 2 February 2022].

Palanov, N. (2016) Vulnerability Categories and Severity Levels: "Informational" Vulnerabilities vs. True Vulnerabilities. Available from: https://www.rapid7.com/blog/post/2016/12/15/vulnerability-categories-and-severity-levels-informational-vulnerabilities-vs-true-vulnerabilities/ [Accessed 13 February 2022].

PortSwigger. (n.d.a) Suspicious input transformation (reflected). Available from: https://portswigger.net/kb/issues/00400d00_suspicious-input-transformation-reflected [Accessed 14 February 2022].

PortSwigger. (n.d.b) Hidden HTTP 2. Available from: https://portswigger.net/kb/issues/01000500_hidden-http-2 [Accessed 14 February 2022].

PortSwigger. (n.d.c) User agent-dependent response. Available from: https://portswigger.net/kb/issues/00400120_user-agent-dependent-response [Accessed 14 February 2022].

PortSwigger. (n.d.d) TLS certificate. Available from: https://portswigger.net/kb/issues/01000100_tls-certificate [Accessed 14 February 2022].

PortSwigger. (n.d.e) Input returned in response (reflected). Available from: https://portswigger.net/kb/issues/00400c00_input-returned-in-response-reflected [Accessed 14 February 2022].

PwC. (2021) Vulnerability Management- Why managing software vulnerabilities is business critical – and how to do it efficiently and effectively. Web: PwC.

Sagar, D., Kukreja, S., Brahma, J., Tyagi, S. & Jain, P. (2018) Studying Open Source Vulnerability Scanners for Vulnerabilities in Web Applications. IIOAB Journal 9(2): 43-49.

Seo, K. (2019) A Methodology for Assessing Security Vulnerability of Cloud Services. *International Journal of Reliable Information and Assurance* 7(2): 1-6. DOI: https://gvpress.com/journals/IJRIA/vol7_no2/vol7_no2_2019_01.html

Tenable. (2017a) E-mail address disclosure. Available from: https://www.tenable.com/plugins/was/98078 [Accessed 14 February 2022].

Tenable. (2017b) Cookie set for parent domain. Available from: https://www.tenable.com/plugins/was/98062 [Accessed 14 February 2022].

Tenable. (2018a) Web Server robots.txt Information Disclosure. Available from: https://www.tenable.com/plugins/nessus/10302 [Accessed 14 February 2022].

Tenable. (2019a) Web Server HTTP Header Information Disclosure. Available from: https://www.tenable.com/plugins/nessus/88099 [Accessed 14 February 2022].

Tenable. (2021a) Cross-Site Request Forgery. Available from: https://www.tenable.com/plugins/was/98112 [Accessed 14 February 2022].

Tenable. (2021b) Cookie Without SameSite Flag Detected. Available from: https://www.tenable.com/plugins/was/115540 [Accessed 14 February 2022].

Tenable. (2021c) Missing HTTP Strict Transport Security Policy. Available from: https://www.tenable.com/plugins/was/98056 [Accessed 14 February 2022].

Tenable. (2021d) Cross-Site Scripting (XSS). Available from: https://www.tenable.com/plugins/was/98104 [Accessed 14 February 2022].

Tenable. (2021e) Missing 'Cache-Control' Header. Available from: https://www.tenable.com/plugins/was/112553 [Accessed 14 February 2022].

Tenable. (2021f) Missing 'X-Frame-Options' Header. Available from: https://www.tenable.com/plugins/was/98060 [Accessed 14 February 2022].

Tenable. (2021g) Password field with auto-complete. Available from: https://www.tenable.com/plugins/was/98081 [Accessed 14 February 2022].

Tenable. (2021h) Mixed Resource Detection. Available from: https://www.tenable.com/plugins/was/98091 [Accessed 14 February 2022].

Tenable. (2021i) Backup file. Available from: https://www.tenable.com/plugins/was/98074 [Accessed 14 February 2022].

Tenable. (2021j) Directory Listing. Available from: https://www.tenable.com/plugins/was/98084 [Accessed 14 February 2022].

Tenable. (2021k) Cookie Without HttpOnly Flag Detected. Available from: https://www.tenable.com/plugins/was/98063 [Accessed 14 February 2022].

Tenable. (2021l) Cookie Without Secure Flag Detected. Available from: https://www.tenable.com/plugins/was/98064 [Accessed 14 February 2022].

Tenable. (2022) Sensitive File Disclosure. Available from: https://www.tenable.com/plugins/nessus/121041 [Accessed 14 February 2022].

Votipka, D., Stevens, R., Redmiles, E., Hu, J. & Mazurek, M. (2018) 'Hackers vs. Testers: A Comparison of Software Vulnerability Discovery Processes', *2018 IEEE Symposium on Security and Privacy (SP)*. San Francisco, California, 20-24 May. New Jersey: IEEE. 374-391.

ZAP Dev Team. (n.d.a) Charset Mismatch. Available from: https://www.zaproxy.org/docs/alerts/90011/ [Accessed 14 February 2022].

ZAP Dev Team. (n.d.b) Application Error Disclosure. Available from: https://www.zaproxy.org/docs/alerts/90022/ [Accessed 14 February 2022].

ZAP Dev Team. (n.d.c) Timestamp Disclosure. Available from: https://www.zaproxy.org/docs/alerts/10096/ [Accessed 14 February 2022].

ZAP Dev Team. (2022a) ZAPping the OWASP Top 10 (2021). Available from: https://www.zaproxy.org/docs/guides/zapping-the-top-10-2021/ [Accessed 10 February 2022].

ZAP Dev Team. (2022b). Available from: https://www.zaproxy.org/docs/alerts/10015/ [Accessed 14 February 2022].

Appendix A

Vector Strings for CVSS 3.1 Calculations

As a way of explaining vulnerabilities, CVSS makes use of what is known as a vector string- a shorthand representation of all the characteristics which influence a vulnerability's score. The values below can be copied and pasted into the CVSS calculator (FIRST, 2021b) to obtain more details about how the vulnerability can be defined, along with its impacts.

ID	Vector String	CVSS Score	Severity	Source
G1	CVSS:3.1/AV:N/AC:H/PR:L/UI:R/S:C/C:H/I:H/A:H	8	High	Self Calculated
G2	CVSS:3.1/AV:N/AC:H/PR:H/UI:R/S:U/C:H/I:H/A:L	6	Medium	Self Calculated
G3	CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:U/C:L/I:L/A:L	5.7	Medium	Self Calculated
B1	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:L/A:N	4.3	Medium	(Tenable, 2021a)
B2	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N	3.1	Low	(Tenable, 2021g)
B3	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N	3.1	Low	(PortSwigger, n.d.c)
B4	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N	0	None	(PortSwigger, n.d.e)
B5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N	0	None	(PortSwigger, n.d.a)
B6	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N	5.3	Medium	(Tenable, 2021i)
B7	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N	0	None	(Tenable, 2017a)
B8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N	0	None	(Tenable, 2018a)
B9	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N	3.7	Low	(Tenable, 2021e)
B10	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N	0	None	(PortSwigger, n.d.d)
B11	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N	6.5	Medium	(Tenable, 2021h)
B12	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N	0	None	(PortSwigger, n.d.b)
Z1	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N	6.1	Medium	(Tenable, 2021d)
Z2	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N	0	None	(ZAP Dev Team, n.d.b)
Z3	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N	5.3	Medium	(Tenable, 2021j)
Z4	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:L/A:N	4.3	Medium	(Tenable, 2021a)
Z5	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N	3.1	Low	(Tenable, 2021k)
Z6	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N	3.1	Low	(Tenable, 2021l)
Z7	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N	3.1	Low	(Tenable, 2021b)

Z8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N	0	None	(Mitre, n.d.a)
Z9	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N	3.7	Low	(Tenable, 2021e)
Z10	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N	0	None	(ZAP Dev Team, n.d.b)
Z11	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N	5.3	Medium	(Tenable, 2019a)
Z12	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N	0	None	(ZAP Dev Team, n.d.c)
Z13	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N	0	None	(Tenable, 2022)
Z14	CVSS:3.1/AV:N/AC:H/PR:H/UI:N/S:U/C:N/I:L/A:N	2.2	Low	(ZAP Dev Team, n.d.a)
R1	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:L/A:N	4.3	Medium	(Tenable, 2021a)
R2	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N	6.5	Medium	(Tenable, 2021c)
R3	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N	3.1	Low	(Tenable, 2021g)
R4	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N	0	None	(Tenable, 2022)
R5	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:N/I:L/A:N	3.1	Low	(Tenable, 2021f)
R6	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N	3.1	Low	(Tenable, 2021l)
R7	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N	3.1	Low	(Tenable, 2021k)
R8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N	0	None	Self Assessment
R9	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N	0	None	(Tenable, 2017b)