Pertanika Proceedings 1 (3): 39 - 44 (2025)

PERTANIKA JOURNALS

PERTANIKA PROCEEDINGS

Journal homepage: http://www.pertanika.upm.edu.my/

Website Security Using Next.js: An Analytical Approach with OWASP Top 10

Raditya Muhammad*, Reihan Manzis Syahputra and Mochamad Iqbal Ardimansyah

Department of Software Engineering, Cibiru Regional Campus, Universitas Pendidikan Indonesia, Bandung 40393, Indonesia

ABSTRACT

This research investigates the effectiveness of Next.js in enhancing web application security, focusing on the IdeaBox Multi-Tenant platform, an idea and innovation collaboration tool. The study adopts an analytical approach referencing OWASP Top 10 and employs Zed Attack Proxy (ZAP) tools, such as Ajax spider and active scan for vulnerability testing. The objective is to measure the impact of Next.js features on mitigating security risks. Initial testing identified 12 security vulnerabilities, including Broken Access Control, Cryptographic Failures, and Security Misconfigurations. After implementing Next.js features like Server-Side Rendering, Middleware, Formik, Yup Validation, Nookies, and bcrypt, the vulnerabilities were reduced to seven categories, with significant risk reduction. The results highlight how Next.js features enhance input validation, data protection, and overall security. This study demonstrates that integrating Next.js effectively minimizes vulnerabilities and improves website security, offering practical guidance for developers. The research underscores the importance of advanced frameworks in mitigating cyber threats and safeguarding sensitive user data.

Keywords: Cyber attacks, OWASP top 10, website security

INTRODUCTION

Website applications, providing essential services. However, as their importance grows, so does the threat to the security of data they handle (Hassan et al., 2023). With web applications often storing sensitive information, robust security testing becomes crucial to prevent cyber attacks (Fata, 2023; Ghelani, 2023; Risky & Yuhandri, 2021).

ARTICLE INFO

Article history:
Received: 30 April 2025
Published: 17 July 2025

DOI: https://doi.org/10.47836/pp.1.3.008

E-mail addresses:
radityamuhammad@upi.edu (Raditya Muhammad)
reihanmanzis@upi.edu (Reihan Manzis Syahputra)
iqbalardimansyah@upi.edu (Mochamad Iqbal Ardimansyah)
* Corresponding author

e-ISSN: 3083-9475

The rise of cyber attacks aimed at compromising data integrity and confidentiality poses significant operational risks (Febryanti et al., 2023).

The IdeaBox platform is vulnerable to common web application threats. To mitigate these risks, the adoption of advanced web development frameworks, such as Next.js, is essential (Lazuardy & Anggraini, 2022). Next.js, a React-based framework, enables the creation of responsive, fast, and secure web applications through features like server-side rendering (Jartarghar et al., 2022), dynamic routing, and Middleware (Dinku, 2022).

Problem Statement

Cyber attacks targeting data integrity and confidentiality pose operational risks (Willberg, 2019), particularly for IdeaBox, which stores sensitive data. While frameworks like Next.js offer advanced security capabilities, there is limited research on their effectiveness in addressing common vulnerabilities identified in the OWASP Top 10 framework (OWASP, 2021; Abdan, 2022). This lack of focused analysis leaves a gap in understanding how Next.js impacts web application security.

Research Question

This study seeks to address the following questions: How do Next.js features impact the application's security level? Finally, to what extent can the integration of Next.js features reduce risks associated with prevalent web application threats? These questions aim to provide a comprehensive understanding of the role of Next.js in enhancing web application security.

METHODOLOGY

The research was conducted in four stages. First, a website domain was selected for testing. Second, the IdeaBox Multi-tenant application, built with React.js and Laravel, was tested using Zed Attack Proxy (ZAP) (Kalaani, 2023) with AJAX Spider and Active Scan methods to identify security vulnerabilities. Third, Next.js features, such as Server-Side Rendering, Middleware, Formik, Yup, Nookies, and bcrypt, were implemented to enhance security. Finally, the application was retested using ZAP's Automated Scan to assess the impact of these Next.js features on reducing security vulnerabilities.

RESULTS AND DISCUSSION

The IdeaBox Multi-Tenant application implemented Next.js features and relevant libraries, including server-side rendering, middleware, Formik, Yup Validation, Nookies, and bcrypt. Security testing was conducted using OWASP guidelines and Zed Attack Proxy (ZAP) tools, employing Ajax Spider and Active Scan methods to identify vulnerabilities and threats as shown in Table 1.

Table 1
Vulnerability after implementing Next.js features

No	Vulnerability Name	Risk level	Qty
1	Content Security Policy (CSP) Header Not Set	Medium	1
2	Missing Anti-Clickjacking Header	Medium	1
3	Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Low	1
4	X-Content-Type-Options Header Missing	Low	33
5	Information Disclosure-Suspicious Comments	Informational	89
6	Modern Web Applications	Informational	1
7	User Agent Fuzzer	Informational	132

The vulnerability analysis is based on the OWASP Top 10 method (Table 2). In the Broken Access Control category, vulnerabilities like server information disclosure via “X-Powered-By” HTTP headers and suspicious source code comments were noted, potentially aiding attackers. The Cryptographic Failures category further highlighted risks of sensitive data exposure. Security Misconfiguration vulnerabilities, including missing headers (CSP, Anti-clickjacking, X-Content-Type-Options), increased risks of XSS attacks, Clickjacking, and content type manipulation, underscore the need for mitigation strategies aligned with NIST’s CVSS framework (Mell et al., 2006).

Table 2
OWASP top 10 vulnerability categorization on testing after next.js implementation

OWASP	Categories	Name	Qty
A01	Broken Access Control	Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s) (1 detected) Information Disclosure-Suspicious Comments (89 detected)	2
A02	Cryptographic Failures	Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s) (1 detected) Information Disclosure-Suspicious Comments (89 detected)	2
A05	Security Misconfiguration	Content Security Policy (CSP) Header Not Set (1 detected), Missing Anti-clickjacking Header (1 detected), X-Content-Type-Options Header Missing (33 detected)	3

The evaluation in Table 3 reveals that the IdeaBox Multi-Tenant application still faces moderate-risk vulnerabilities, including Broken Access Control (CVSS 5.1), Cryptographic Failures (5.2), and Security Misconfiguration (6.4). While Next.js features have reduced some risks, exploitable gaps remain, necessitating additional security measures, continuous monitoring, and further integration with complementary security technologies.

Table 3
Vulnerability risk level after implementing Next.js features

No	Vulnerability Category	CVSS Score	Status
1	Broken Access Control	5.1	Medium
2	Cryptographic Failures	5.2	Medium
3	Injection	N/A	N/A
4	Insecure Design	N/A	N/A
5	Security Misconfiguration	6.4	Medium
6	Vulnerable and Outdated Components	N/A	N/A
7	Identification and Authentication Failures	N/A	N/A
8	Software and Data Integrity Failures	N/A	N/A
9	Security Logging and Monitoring Failures	N/A	N/A
10	Server-Side Request Forgery (SSRF)	N/A	N/A

Table 4 shows that after implementing Next.js security features, vulnerabilities in the IdeaBox Multi-tenant website vary in likelihood, with Broken Access Control scoring low (2,875), and Cryptographic Failures and Security Misconfiguration scoring medium (4,375 and 5,750).

Table 4
Likelihood level after implementing Next.js features

No	Vulnerability Category	Likelihood Score	Likelihood Level
1	Broken Access Control	2,875	Low
2	Cryptographic Failures	4,375	Medium
3	Injection	N/A	N/A
4	Insecure Design	N/A	N/A
5	Security Misconfiguration	5,750	Medium
6	Vulnerable and Outdated Components	N/A	N/A
7	Identification and Authentication Failures	N/A	N/A
8	Software and Data Integrity Failures	N/A	N/A
9	Security Logging and Monitoring Failures	N/A	N/A
10	Server-Side Request Forgery (SSRF)	N/A	N/A

The implementation of Next.js features reduced the IdeaBox Multi-Tenant application's risk level from high (7.9) to medium (5.57), with a significant drop in vulnerabilities. Low-risk findings decreased from 246 to 34, medium-risk findings from 248 to 2, and no high-risk vulnerabilities remained. OWASP Top 10 issues like Broken Access Control and Security Misconfiguration were significantly mitigated, and Identification and Authentication Failures were eliminated. These results demonstrate the effective use of Next.js features in enhancing application security.

CONCLUSION

The implementation of Next.js significantly improved the security of the IdeaBox Multi-Tenant application, reducing vulnerabilities from 12 to 7 categories with lower risks. Features like Server-Side Rendering, Middleware, Formik, Yup Validation, Nookies, and bcrypt effectively addressed data handling, access control, input validation, and encryption. The OWASP Top 10 framework and CVSS were instrumental in assessing risks. While security was enhanced, some gaps remain, requiring ongoing monitoring and further mitigation.

ACKNOWLEDGEMENT

We sincerely appreciate the support and provision of essential resources from Universitas Pendidikan Indonesia, which significantly contributed to the development of this project.

REFERENCES

Abdan, M. K. (2022). *Web-Based Information System Security Testing Based on the OWASP WSTG V4.2 Framework (Case study: Sekawan System V1)*. Universitas Islam Indonesia.

Dinku, Z. (2022). *React.js vs. Next.js* [Unpublish bachelor's thesis]. Metropolia University of Applied Sciences.

Fata, D. (2023). *Evaluation of Security Vulnerability Risk Using OWASP Methodology on Academic Information System Web Applications UIN AR-RANIRY*. Universitas Islam Negeri Ar-Raniry.

Febryanti, P. C., Subartini, B., & Riaman, R. (2023). Calculation of cyber risk level in digital financial services based on aggregate loss costs. *FIBONACCI: Jurnal Pendidikan Matematika Dan Matematika*, *9*(1), 95–104.

Ghelani, D. (2022). Cyber security, cyber threats, implication and future perspectives: A review. *American Journal of Science, Engineering and Technology*, *3*(6), 12–19.

Hassan, A. A., Rony, M. A., Yuliazmi, & Putra, B. C. (2023). E-commerce implementation using content management system (CMS) at Linda Collection Store. *2nd Seminar Nasional Mahasiswa Fakultas Teknologi (Senafti), 2*(1), 659–667.

Jartarghar, H. A., Salanke, & Dalali, S. (2022). React Apps with server-side rendering: Next.js. *Journal of Telecommunication, Electronic and Computer Engineering*, *14*(4), 25–29. https://doi.org/10.54554/jtec.2022.14.04.005

Kalaani, C. (2023). *OWASP ZAP vs Snort for SQLi Vulnerability Scanning*. Georgia Southern University.

Lazuardy, M. F. S., & Anggraini, D. (2022). Modern front end web architectures with React.Js and Next.Js. *International Research Journal of Advanced Engineering and Science*, *7*(1), 132–141.

Mell, P., Scarfone, K., and Romanosky, S. (2006). Common vulnerability scoring system. *IEEE Security & Privacy, 4*(6), 85-89. https://doi.org/10.1109/MSP.2006.145

OWASP. (2021). *OWASP Top 10:2021*. Open Web Application Security Project. https://owasp.org/Top10/

Risky, M. A. Z., & Yuhandri, Y. (2021). Optimization in website security penetration testing using SQL injection and XSS techniques. *Jurnal Sistim Informasi dan Teknologi, 3*(4), 215–220. https://doi.org/10.37034/jsisfotek.v3i4.68

Willberg, M. (2019). *Web application security testing with owasp top 10 framework* [Unpublished bachelor's thesis]. Turku University of Applied Science.