Analysis of Vulnerabilities and Consequences of Missing Hardening in WordPress-Based CMS Environments

P. KABATA

pawel.kabata@artixen.net

Artixen.net Sp. z o.o.
Kowieńska Str. 22/156, 03-438 Warsaw, Poland

WordPress is currently the most widely used content management system and constitutes a key component of the modern web application ecosystem. Its evolution from a blogging platform into a universal CMS has significantly expanded its functionality while simultaneously increasing architectural complexity and the attack surface. The open-source nature of the project, the extensive ecosystem of plugins and themes, and a simplified permission model make the security of WordPress installations largely dependent on the quality of external components and the configuration of the hosting environment. This paper provides a review of the most common vulnerabilities observed in WordPress-based environments and discusses hardening techniques including update management, permission restrictions, authentication strengthening, and security monitoring. The analysis indicates that the absence of hardening measures leads to an increased risk of unauthorized access, malware infections, and operational and reputational losses, confirming the necessity of treating WordPress hardening as a continuous process.

Keywords: WordPress security, CMS, hardening.

DOI: 10.5604/01.3001.0055.5705

1. Introduction

WordPress is currently the most widely used content management system on the Internet and, according to various estimates, accounts for approximately 40% to 60% of all websites operating online. However, its origins differ from the role it fulfills today. The system was initially developed as a blogging platform designed for ease of content publishing and intuitive use, and later gained widespread popularity due to its flexibility, scalability, and extensibility. As a result, WordPress is also employed in use cases that were not part of its original design objectives, which naturally leads to a situation in which a portion of its functionality is implemented through external components rather than the system core. These extensions take the form of plugins and themes, whose quality, update frequency, and level of security vary, thereby influencing the overall risk profile of a given installation [2], [10].

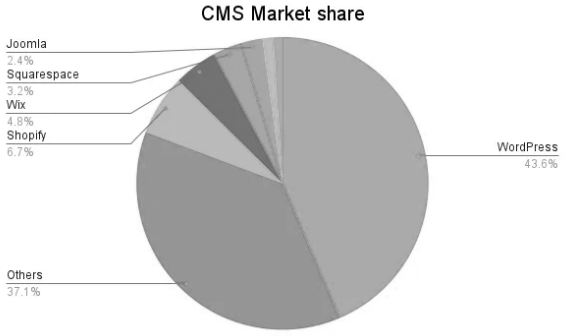

Fig. 1. CMS Market Share
Source: wpmet.com [15]

The popularity of WordPress directly translates into an expanded attack surface. As the dominant CMS, it represents an attractive target for automated vulnerability scanning campaigns and large-scale system compromise attempts. Industry reports indicate that security incidents affecting CMS environments very frequently involve WordPress installations, with the underlying causes typically being a combination of outdated components, configuration errors, and vulnerabilities in third-party extensions [3], [4], [5], [6].

The open-source nature of the project promotes transparency and facilitates rapid

vulnerability discovery; however, it does not guarantee security. In practice, open-source projects may still contain implementation flaws, and the extension ecosystem can include low--quality components, abandoned plugins, or software developed without formal security audits. Consequently, the security of WordPress should be regarded as a composite outcome of the quality of the system core, third-party extensions, and the configuration of the hosting environment [2], [3], [4].

In this context, hardening, understood as a set of configuration and administrative measures aimed at reducing the attack surface, becomes a fundamental security mechanism. It includes update management, restriction of privileges, strengthening of authentication mechanisms, control of external components, and security event monitoring. The objective of this paper is to review key threats affecting WordPress environments, discuss hardening techniques, and analyze the consequences resulting from the absence of such measures in production systems [1], [7], [10], [11].

2. WordPress Architecture and Security Model

WordPress is a system based on a modular architecture in which four main components play a fundamental role: WordPress Core, themes, plugins, and the database layer. This structure provides a high degree of flexibility and scalability; however, it simultaneously introduces security-related challenges. A significant portion of the functionality of typical WordPress installations does not originate from the system core but from extensions developed by third-party entities, which means that the practical security of an installation is often strongly dependent on their quality and maintenance [2], [10], [12].

WordPress Core is developed continuously and is subject to regular updates. Vulnerabilities within the core occur less frequently than those found in extensions; nevertheless, their impact may be substantial due to the wide distribution of a given release. At the same time, in practice, the security of the core can be weakened by improper server configuration or by the introduction of vulnerable code within themes and plugins that integrate directly with the system's mechanisms [2], [3], [4], [12].

An important component of the WordPress security model is the user permission model.

Tab. 1. Table of WordPress versions in use [16]

Version	Usage
6.9	50.88%
6.8	23.01%
6.7	5.31%
6.6	2.48%
6.5	1.90%
6.4	1.94%
6.3	0.96%
6.2	1.39%
6.1	1.29%
6.0	1.00%
5.9	0.78%
5.8	0.96%
5.7	0.82%
5.6	0.53%
5.5	0.73%
5.4	0.84%
5.3	0.67%
5.2	0.63%
5.1	0.42%
5.0	0.25%
4.9	1.28%
4.8	0.33%
4.7	0.40%
4.6	0.15%
4.5	0.15%
4.4	0.15%
4.3	0.10%
4.2	0.11%
4.1	0.09%
4.0	0.06%
3.9	0.07%
3.8	0.04%
3.7	0.01%
3.6	0.05%
3.5	0.06%
3.4	0.04%
3.3	0.03%
3.2	0.02%
3.1	0.02%
3.0	0.05%

System roles are defined in a relatively simplified manner, which stems from the historical design assumptions of the platform. The lack of granular access control over individual functions and modules in a standard installation means that the compromise of a high-privilege account has particularly severe consequences, as it enables an attacker to extensively interfere with configuration, content, and application

components. In practice, more advanced access control mechanisms are often implemented through plugins, which once again shifts part of the responsibility for security to external components [2], [10].

Another factor affecting security is the REST API. While it facilitates integrations and programmatic access, it can also constitute an additional attack vector, particularly when it is not actively used in a given deployment. As part of hardening efforts, it is often justified to limit the exposure of these interfaces to the minimum necessary, in accordance with the principle of reducing the attack surface [1], [7].

The security of WordPress is also dependent on the hosting environment. A typical installation relies on a web server, the PHP interpreter and a database management system, and the available security mechanisms depend on the configuration and the level of control provided to the administrator by the hosting provider. The ability to apply server-level rules, enforce directory separation, access security-related settings and deploy protective mechanisms determines whether hardening can be effectively implemented in practice or remains merely a set of theoretical recommendations. In this context, practical guidelines described in the literature on securing and recovering WordPress-based websites, as well as comparative analyses involving other CMS platforms, are particularly relevant, as they allow WordPress to be better positioned within the broader landscape of similar solutions [10].

3. Common Vulnerabilities in WordPress Installations

The security of WordPress depends on the interaction of multiple factors, including the system architecture, the quality of deployed extensions, the type of server environment, and administrative practices. Vulnerabilities may arise both from flaws in the system core and from outdated plugins, unverified themes, weaknesses in authentication mechanisms, or improper infrastructure configuration. The complexity of the WordPress ecosystem results in a multi-layered threat landscape, and effective protection requires an understanding of common vulnerability categories and their sources [2], [3], [4], [5], [14].

Vulnerabilities in the WordPress core occur relatively infrequently, which is a consequence of the development process, the public nature of the source code, and regular updates. However, when a vulnerability does occur in the core, its consequences can be significant, as it affects all installations based on a given version that have not been updated. From a security perspective, rapid deployment of patches is therefore critical, as is the understanding that even rare core vulnerabilities may have a wide impact on a global scale [2], [3].

Plugins constitute the largest attack surface in WordPress installations and are responsible for a substantial proportion of known security incidents. They are developed by numerous independent entities, and their quality, maintenance practices, and update frequency vary considerably. Reports on threats and vulnerabilities in the WordPress ecosystem consistently indicate that the dominant share of security issues originates from extensions rather than from the system core itself. In practice, plugin vulnerabilities include input validation flaws, privilege escalation issues, vulnerabilities leading to remote code execution, and integration-related errors. A particularly serious problem is posed by plugins abandoned by their developers, which do not receive updates in line with the evolution of WordPress, as well as components obtained from unofficial sources [3], [4], [5], [9], [13].

Themes, which are responsible for the visual layer and part of the user-facing functionality, also represent a significant risk factor. They may include outdated libraries, including JavaScript components, or implement custom data presentation mechanisms that are vulnerable to code injection attacks. Additional risks arise from themes obtained from unverified sources, which may contain hidden mechanisms enabling further compromise of the environment. These phenomena are discussed in security analyses of the WordPress ecosystem and in materials addressing vulnerabilities in themes and plugins, allowing themes to be incorporated into a comprehensive classification of risk sources, [3], [4], [8], [13].

Login mechanisms are a common target of automated attacks, including large-scale brute-force attempts carried out by bots. The absence of additional protective measures, such as multi-factor authentication, limitations on the number of login attempts, or monitoring of anomalous login patterns, increases the likelihood of account compromise. In practice, the takeover of an account with extensive privileges is particularly critical, as it enables the installation of malicious plugins, modification of configuration settings, and the persistent deployment of access maintenance mechanisms.

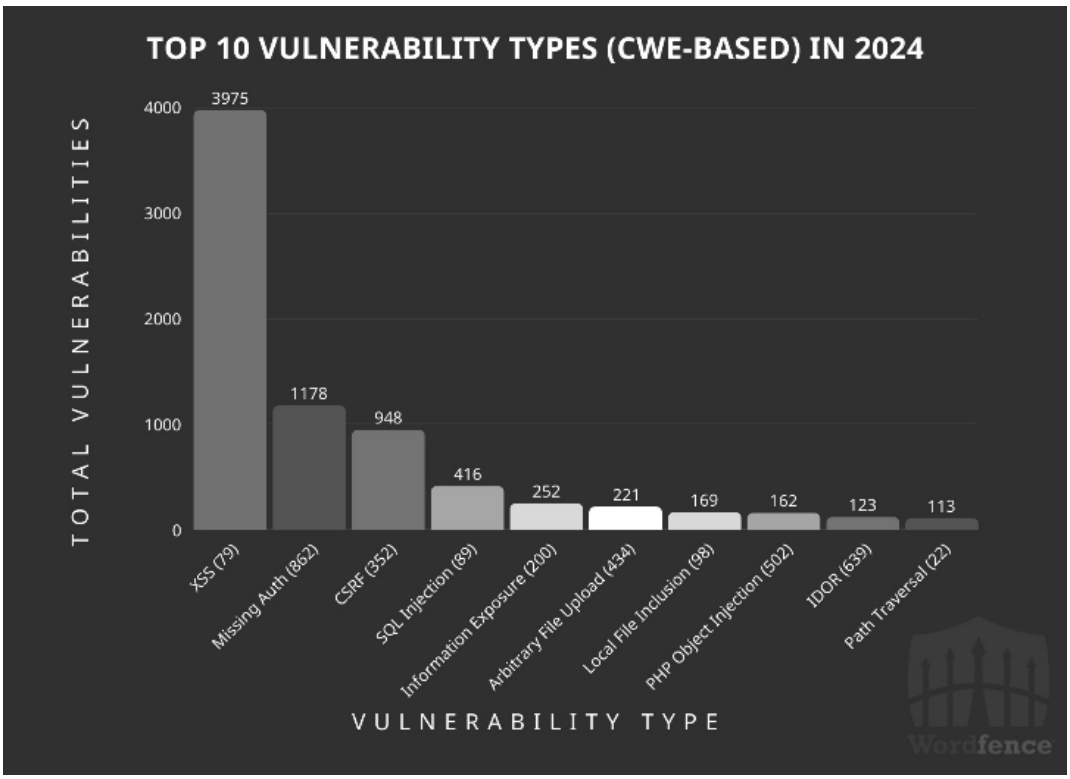

Fig. 2. Top 10 Vulnerabilities Disclosed in 2024
by CWE ID [5]

In this area, both industry reports on real--world attack trends and literature on web application security are valuable, as they provide a conceptual framework for describing these mechanisms [5], [7], [11], [12].

Improper configuration of the server environment may result in a situation where even a relatively well-maintained WordPress installation remains vulnerable to attacks stemming from infrastructural weaknesses. Typical issues include overly permissive file permissions, exposure of sensitive resources, misconfiguration of network services, lack of directory separation between applications, and leaving active components that increase the attack surface, such as XML-RPC in scenarios where it is not required. In practice, these configuration errors often act as enabling factors for persistent compromise following the exploitation of a vulnerability in an extension. These observations are consistent with both the official WordPress hardening documentation and practical guidelines concerning the securing and recovery of WordPress-based websites [1], [7], [10].

4. WordPress Hardening Techniques

WordPress hardening comprises a set of practices aimed at reducing system vulnerabilities through proper update management, environment configuration, access control, and activity monitoring. Effective hardening is a preventive and process-oriented approach, as the WordPress environment evolves alongside updates to the core, plugins, and themes, while the threat landscape continuously changes with the emergence of new attack techniques. Recommendations in this area are formulated both by the WordPress developers and by specialized entities analyzing threats within the WordPress ecosystem [1], [2], [7].

Regular updates of the core, plugins, and themes constitute one of the most important elements of WordPress protection. Reports on website security threats indicate that delays in applying updates increase the risk of exploiting known vulnerabilities, particularly when attacks are automated and rely on publicly available information about security flaws. In practice, automating core updates may be acceptable, especially for stable releases; however, compatibility risks should be taken into account. An essential complement to the update process is the creation of backups, preferably stored off-server, to ensure system recovery after an incident or a failed update [3], [5], [10].

Proper configuration of file and directory permissions is critical for limiting the impact of a potential security breach. The web server should have only the minimum access required for application operation, and critical resources should not be writable unless strictly necessary.

In practice, additional restrictions on script execution within specific directories are often applied, reducing the possibility of persistent malware deployment following access to upload mechanisms. Directory separation between applications hosted on the same server further limits the risk of escalation from a compromised instance to the entire environment. This approach is consistent with WordPress hardening documentation and practical guidelines on securing WordPress installations [1], [10].

Protecting the login process is one of the key components of hardening. The use of multi-factor authentication and password policies significantly reduces the risk of account compromise. These measures are complemented by limiting login attempts, blocking suspicious IP addresses, and monitoring login events to identify automated brute-force campaigns. Industry reports provide empirical data on the scale of such attacks, while web application security literature places these measures within a broader protection model [5], [7], [11].

WordPress hardening also requires actions at the server and infrastructure level. These include disabling unused functionalities and applying server-level rules to reduce the attack surface. In practice, limiting or disabling XML-RPC in scenarios where it is not required is often recommended, along with the use of traffic filtering mechanisms. Depending on the capabilities of the hosting environment, an application firewall (WAF) or server-side WAF service may play a significant role by blocking suspicious requests and attempts to exploit known attack vectors. In this area, both WordPress documentation and threat and incident reports within the ecosystem are particularly useful [1], [5], [6], [7], [13].

Effective WordPress protection requires monitoring mechanisms that enable rapid detection of anomalies. Log analysis, monitoring of login attempts, file integrity checks, and the use of scanning tools form the foundation of early threat detection. Industry reports emphasize the importance of incident detectability and highlight that compromises may persist without visible symptoms if monitoring is not performed systematically. In practice, monitoring serves as the link between preventive protection and incident response, helping to limit the overall impact of security breaches [5], [6], [9], [10].

5. Consequences of Missing WordPress Hardening

The absence of adequate security measures in WordPress installations leads to a significant increase in the risk of security breaches and incidents affecting data integrity, availability, and confidentiality. Neglecting hardening increases the effectiveness of automated attacks, facilitates compromise through vulnerable extensions, and enables attackers to maintain persistence within the environment. The consequences include technical, operational, and reputational impacts [5], [6], [10], [12].

The most direct consequence of missing hardening is an increased risk of user account compromise, including administrator accounts. The absence of multi-factor authentication, weak passwords, and a lack of mechanisms limiting login attempts facilitate account takeovers, after which attackers can modify configuration settings and install malicious components. Reports indicate that password-guessing attempts and mass attacks against login panels are widespread phenomena [5], [7], [11].

The lack of file permission restrictions, the use of outdated plugins, and the absence of monitoring significantly increase the risk of malware infections. Consequences may include website defacement, redirection of users to malicious domains, installation of backdoors, and the use of the website for further malware distribution. Analyses of infection trends indicate that CMS website compromises are often the result of neglected updates and configuration errors [5], [6], [9].

Security breaches may result in the leakage of data stored in WordPress databases, including user information and authentication credentials. In practice, such leaks arise from both vulnerabilities in extensions and configuration errors that expose sensitive files and resources. The consequences include legal and financial implications as well as the need for remediation actions. Practical guides on securing and recovering WordPress-based websites are particularly relevant in this context, as they describe typical breach scenarios and mitigation strategies [10], [11].

Compromised WordPress installations are often exploited to inject SEO spam, hidden links, or malicious redirects. As a result, domain credibility declines, organic traffic is lost, and the website may be flagged as unsafe. Malware trend reports indicate that such activities are a frequent component of campaigns conducted on compromised websites [6], [5].

Following a compromise, attackers may leverage server resources as part of a botnet. This leads to increased infrastructure load, degraded performance, potential hosting suspension, and the risk of participation in distributed denial-of-service attacks. This relationship is evident in incident analyses and in real-world response practices, where website compromise often results in secondary abuse [14], [6].

The consequences of missing hardening extend beyond technical aspects. Security breaches negatively affect the reputation of website owners, reduce user and partner trust, and may result in financial losses caused by downtime and recovery costs. Remediation activities typically involve incident analysis, environment cleanup, data restoration, and the deployment of security measures that could have been applied earlier as part of the hardening process [10], [5].

6. Comparison of Hardened and Non-Hardened Environments

The effectiveness of hardening is clearly observable when comparing environments in which protective practices have been implemented with installations maintained without such mechanisms. This comparison highlights differences in attack surface size, resistance to common automated campaigns, incident detectability, and the scale of post-compromise consequences [1], [7], [5].

In non-hardened environments, the attack surface is larger due to the presence of outdated plugins, themes, and components, as well as unrestricted exposure of services and interfaces. In hardened environments, the attack surface is reduced through updates, minimization of unnecessary functionalities, restriction of access to administrative interfaces, and control of external components [1], [3], [4], [7].

Non-hardened environments are more susceptible to brute-force attacks, exploitation of known extension vulnerabilities, and mass scanning activities. Hardened environments, through strengthened authentication and login attempt limitations, hinder such attacks, while additional filtering mechanisms and server-level rules may block exploitation attempts at an early stage [5], [7], [6].

The absence of monitoring in non-hardened environments often results in incident detection only after visible effects occur, such as defacement or performance degradation. In hardened environments, monitoring and log analysis increase the likelihood of early incident detection, thereby reducing attacker dwell time and limiting the extent of damage [6], [5], [9].

Compromises in non-hardened environments more frequently lead to service disruptions, slowdowns, and hosting provider blocks. In hardened environments, operational stability is higher, as the risk of infection and infrastructural abuse is reduced and protective mechanisms limit the number of successful exploitation attempts [5], [14].

The costs of incident remediation, data recovery, and trust rebuilding are significantly higher in the absence of hardening. In hardened environments, costs are more predictable and primarily associated with maintaining updates, monitoring, and periodic reviews. This comparison leads to the conclusion that hardening is not only a technical protection measure but also a tool for minimizing operational and reputational losses [5], [6], [10].

7. Conclusions

The analysis presented in this paper confirms that the security of WordPress installations is significantly dependent on the quality of implemented hardening practices and on informed management of ecosystem components. The greatest risks arise from plugins and themes, whose vulnerabilities constitute the dominant source of security incidents, while the lack of updates and the use of components of uncertain origin further increase susceptibility to compromise [3], [4], [5], [8], [9].

Hardening measures including update management, privilege restriction, authentication strengthening, reduction of unnecessary functionalities, and event monitoring significantly reduce the attack surface and enhance resistance to automated campaigns and post-compromise abuse. At the same time, the consequences of missing hardening extend beyond technical effects, such as malware infections or account takeovers, to include reputational and financial impacts. This justifies treating WordPress hardening as a continuous process rather than a one-time configuration activity [1], [7], [10], [11].

In light of the growing number of threats within the WordPress ecosystem and the ongoing evolution of extensions, future work may focus on more formalized risk assessment methods for specific sets of plugins and themes, automation of security testing prior to update deployment, and auditing procedures for third-party components in production environments [4], [5], [14].

From the author's perspective, grounded in direct involvement in the design, deployment, and maintenance of CMS-based systems, the presented analysis reflects not only a theoretical review but also practical observations derived from real-world implementations. Long-term exposure to WordPress-based environments indicates that, contrary to common criticism, the platform has significantly matured over time in terms of code quality, performance optimization, and security mechanisms. This evolution demonstrates that the longevity of a widely adopted platform can contribute positively to its robustness, provided that development and maintenance processes remain active and responsive.

It should be emphasized that no content management system is originally designed with security as its sole or primary objective. Some platforms address security challenges more effectively than others; however, the fundamental assumption across CMS solutions is extensibility rather than complete security by default. This design philosophy explains the existence of numerous third-party extensions, security plugins, and external protection services intended to enhance baseline security. WordPress follows this model explicitly, offering a flexible core that can be expanded and reinforced depending on deployment requirements.

Notably, if WordPress were to operate strictly within the scope of its original design assumptions as a blogging platform, using only default themes and built-in functionalities, many of the security issues discussed in this work would likely not occur. This observation holds true particularly when combined with basic hardening practices, such as restricting administrative access and disabling unused features. In practice, however, WordPress is rarely used in such a minimal configuration, which fundamentally alters its risk profile.

The widespread adoption of WordPress also places it somewhat outside the mainstream focus of contemporary cybersecurity research, where attention is increasingly directed toward highly specialized domains, artificial intelligence – driven threats, and enterprise-scale infrastructures. CMS platforms, especially those based on PHP and containing substantial legacy components, are often perceived as less attractive research targets. At the same time, their ubiquity creates a paradox: while individual users can easily build functional websites using visual editors, securing and properly configuring these systems remains a non-trivial task requiring technical expertise.

This gap highlights a clear niche for further research, education, and dissemination of best practices related to CMS security. Given the continued prevalence of WordPress in production environments, there is a strong need to promote awareness of hardening techniques and configuration guidelines, particularly for non-expert administrators. Addressing this need may contribute not only to improving the security posture of individual installations but also to reducing the overall attack surface of the web ecosystem.

8. Bibliography

[1] WordPress Developer Resources, "Hardening WordPress", https://developer.wordpress.org/advanced-administration/security/hardening/ (accessed: 10.10.2025).
[2] WordPress, "WordPress Security White Paper", https://github.com/WordPress/Security-White-Paper/blob/master/WordPressSecurityWhitePaper.pdf (accessed: 10.10.2025).
[3] WPScan, "WPScan 2024 Website Threat Report", https://wpscan.com/2024-website-threat-report/ (accessed: 10.10.2025).
[4] Patchstack, "State of WordPress Security in 2025", https://patchstack.com/whitepaper/state-of-wordpress-security-in-2025/ (accessed: 10.10.2025).
[5] Wordfence, "2024 Annual WordPress Security Report", https://www.wordfence.com/blog/2025/04/2024-annual-wordpress-security-report-by-wordfence/ (accessed: 10.10.2025).
[6] Sucuri, "SiteCheck Malware Trends Report 2024", https://sucuri.net/reports/sitecheck-malware-trends-report-2024/ (accessed: 10.10.2025).
[7] WPScan Blog, "Hardening WordPress: 26 Steps to Harden WP Like an Expert", https://wpscan.com/blog/hardening-wordpress/ (accessed: 10.10.2025).
[8] Kaspersky daily, "Vulnerabilities in WordPress Plugins and Themes", https://www.kaspersky.com/blog/vulnerable-wordpress-plugins-and-themes/54228/ (accessed: 15.10.2025).
[9] TuxCare, "Understanding the RCE Vulnerabilities in WordPress Plugins", https://tuxcare.com/blog/wordpress-plugin-vulnerabilities/ (accessed: 15.10.2025).
[10] Frankowski P., *WordPress i Joomla! Zabezpieczanie i ratowanie stron WWW*,

Helion, Gliwice 2017, ISBN 978-83-283-2899-0.
[11] Sajdak M. (red.), *Bezpieczeństwo aplikacji webowych*, Securitum, Kraków 2019, ISBN 978-83-954853-0-5
[12] Ojemade S., Omede G.C., Abel E., Akazue M., "A Model for a Secured and Reusable Web Content Management System with WordPress", *International Journal of Advances in Engineering and Management*, Vol. 6, Issue 5, 593–603 (2024), https://ijaem.net/issue_dcp/A%20Model%20for%20a%20Secured%20and%20Reusable%20Web%20Content%20Management%20System%20with%20Wordpress.pdf (accessed: 15.10.2025).
[13] Zamościński P., Kozieł Z., "Analysis of security CMS platforms by vulnerability scanners", *JCSI*, Vol. 16, 261–268 (2020), https://ph.pollub.pl/index.php/jcsi/article/view/2020 (accessed: 15.10.2025).
[14] MITRE, CVE Program, https://www.cve.org/CVERecord/SearchResults?query=wordpress (accessed: 15.10.2025).
[15] WPMet, CMS Market Share in 2025: All Latest Trends, Statistics, and Insights, https://wpmet.com/pl/cms-market-share/ (accessed: 10.12.2025).
[16] Table of types of versions running WordPress, https://wordpress.org/about/stats/ (accessed: 10.12.2025).

Analiza podatności i konsekwencji braku hardeningu w środowisku CMS WordPress

P. KABATA

WordPress jest obecnie najczęściej wykorzystywanym systemem zarządzania treścią i stanowi istotny element współczesnego ekosystemu aplikacji webowych. Jego ewolucja z platformy blogowej do uniwersalnego systemu CMS zwiększyła zakres funkcjonalności, lecz jednocześnie doprowadziła do wzrostu złożoności architektury oraz powierzchni ataku. Otwarty charakter projektu, rozbudowany ekosystem wtyczek i motywów oraz uproszczony model zarządzania uprawnieniami sprawiają, że bezpieczeństwo instalacji WordPressa w dużej mierze zależy od jakości komponentów zewnętrznych oraz konfiguracji środowiska serwerowego. W artykule przedstawiono przegląd najczęściej występujących podatności w środowiskach WordPressa oraz omówiono techniki hardeningu obejmujące aktualizacje, ograniczanie uprawnień, wzmocnienie uwierzytelniania i monitorowanie zdarzeń. Analiza wskazuje, że brak hardeningu prowadzi do zwiększonego ryzyka nieautoryzowanego dostępu, infekcji malware oraz strat operacyjnych i reputacyjnych, co potwierdza konieczność traktowania hardeningu jako procesu ciągłego.

Słowa kluczowe: bezpieczeństwo WordPressa, systemy CMS, hardening.