Analysis of Web Application Security: Integrating WAF and SSL/TLS for Enhanced CMS Protection

Salman Shakeel, Arbab Masood Ahmad, Yasir Saleeem Afridi, Rehmat Ullah
Department of Computer Systems Engineering, University of Engineering and Technology, Peshawar
*Correspondence: 20pwcse1925@uetpeshawar.edu.pk, arbabmasood@uetpeshawar.edu.pk, yasirsaleem@uetpeshawar.edu.pk, rehmatullah@uetpeshawar.edu.pk

Citation | Shakeel. S, Ahmad. A. M, Afridi. Y, Ullah. R, "Analysis of Web Application Security: Integrating WAF and SSL/TLS for Enhanced CMS Protection", IJIST, Vol. 7 Issue. 4 pp 2617-2629, November 2025
Received | October 4, 2025 **Revised |** October 19, 2025 **Accepted |** October 25, 2025 **Published |** November 6, 2025.

Content Management Systems (CMS) such as WordPress, Joomla and Drupal power a significant portion of the web, making them prime targets for cyber threats, including TLS downgrade attacks, SQL injection (SQLi), cross-site scripting (XSS) and brute force attempts. Traditional security mechanisms often fail to mitigate these sophisticated attacks, leading to data breaches and unauthorized access. This research implements a multi-layered security framework integrating Web Application Firewalls (WAFs), TLS 1.3 enforcement, AI-driven vulnerability detection and enhanced security headers on a WordPress test environment. Security audits using OWASP ZAP, Nessus and Burp Suite validated the effectiveness of each component. Results demonstrate an 80% reduction in brute force attacks, a 93% decrease in SQL injection attempts and a 100% elimination of XSS vulnerabilities. The implementation of WAF filtering, real-time monitoring and strict access controls significantly reduced the attack surface. This study provides a scalable, adaptive security model capable of evolving with emerging cybersecurity challenges, offering a vital contribution to web application security.

Keywords: WordPress Security, CMS Security, Web Application Security, Brute Force Attack, SQL Injection, Cross-Site Scripting (XSS), DDoS Attacks, Two-Factor Authentication (2FA), Firewall Protection, Intrusion Detection Systems (IDS), Security Plugins, Content Delivery Networks (CDN), Secure HTTP Headers.

Introduction:

Content Management Systems (CMS) have revolutionized web development by providing user-friendly platforms for content creation and management. However, their widespread adoption has made them attractive targets for cybercriminals. According to Sucuri's 2023 Website Security Report, WordPress is the most vulnerable CMS, followed by Joomla and Drupal [1]. The inherent vulnerabilities in these platforms necessitate robust security measures to protect sensitive data and maintain website integrity [2].

Problem Statement Despite the availability of security plugins, periodic updates and security recommendations, many CMS-based websites continue to suffer from severe vulnerabilities. Several security gaps expose these websites to a wide range of cyberattacks, including:

TLS Downgrade Attacks – Cybercriminals take advantage of weak or improperly configured TLS protocols to downgrade encryption standards, allowing them to intercept sensitive information and compromise otherwise secure connections.

SQL Injection (SQLi) – Hackers inject malicious SQL queries into website databases, extracting confidential information such as user credentials, payment details and system configurations [3].

Brute Force Attacks – Automated bots systematically attempt multiple username and password combinations to gain unauthorized access to CMS admin panels. Weak passwords and the absence of multifactor authentication (MFA) increase the risk of account compromise.

Cross-Site Scripting (XSS) – Scripts hidden within web elements may execute to exfiltrate user information, manipulate page content and propagate malicious software. These vulnerabilities underscore the urgent need for a comprehensive, multi-layered security strategy that extends beyond conventional protection measures [4]. Relying solely on basic security plugins and manual security updates is insufficient to combat modern threats. Instead, a layered security approach that combines advanced encryption, AI-powered threat detection, real-time security monitoring and proactive firewall implementations is required to effectively fortify CMS-based websites against evolving cyber risks [5].

Literature Review:

Web application security research has evolved across multiple dimensions, addressing vulnerability identification, protection mechanisms and emerging intelligent detection systems. This review synthesizes key contributions in CMS security and identifies gaps that this research addresses.

CMS Vulnerability Landscape: WordPress, Joomla and Drupal collectively power over 60% of content management systems globally, making them prime targets for exploitation. The authors in reference [2] conducted a comprehensive security analysis of open-source CMS platforms, revealing that 68% of WordPress breaches originate from outdated or vulnerable plugins rather than core software weaknesses. This finding highlights a critical challenge in CMS security: the distributed nature of third-party extensions creates an expanded attack surface that traditional security measures struggle to protect.

The vulnerability landscape extends beyond plugin weaknesses to encompass configuration inconsistencies across platforms. [5] identified significant security header discrepancies between mobile and desktop implementations of the same websites, with 43% of surveyed sites failing to enforce HTTP Strict Transport Security (HSTS) on mobile versions despite implementing it on desktop. These inconsistencies enable session hijacking, TLS downgrade attacks and insecure content loading, demonstrating that even well-configured systems can harbor exploitable vulnerabilities.

Large-scale internet scanning initiatives have revealed the prevalence of cryptographic misconfigurations. Author [6] developed Censys, a search engine backed by comprehensive IPv4 space scanning, which exposed that 31% of surveyed domains still accept TLS 1.0

connections despite documented vulnerabilities. Author [7] extended this work with an internet security scanner focused on TLS adoption trends, demonstrating that configuration drift over time frequently reintroduces previously patched vulnerabilities, underscoring the need for continuous monitoring rather than point-in-time assessments.

Protection Mechanisms and Security Frameworks:

Research into web application protection has progressed from isolated defenses to integrated security frameworks. Author [8] categorized protection approaches into three primary strategies: security by construction (building secure systems from the ground up), security by verification (formally proving system properties) and security by protection (implementing runtime defenses). Their survey emphasizes that effective web security requires layered defenses addressing multiple attack vectors simultaneously.

Web Application Firewalls have emerged as a critical perimeter defense mechanism. The OWASP Foundation's Core Rule Set provides a baseline for detecting SQL injection, cross-site scripting and other common attacks, though effectiveness depends heavily on proper tuning to minimize false positives while maintaining protection efficacy. Cloudflare's 2023 DDoS threat report documented successful mitigation of attacks exceeding 1 Tbps through integrated CDN and WAF architectures, demonstrating the scalability of cloud-based protection mechanisms.

Transport Layer Security has undergone significant evolution with the adoption of TLS 1.3, which eliminates vulnerable cipher suites and reduces handshake latency. However, deployment challenges persist, as evidenced by NIST's 2023 Cybersecurity Framework, noting that legacy system compatibility concerns delay TLS 1.3 adoption in enterprise environments. The implementation of HTTP Strict Transport Security (HSTS) with preloading provides an additional defense layer, though Mendoza et al.'s findings regarding mobile-desktop inconsistencies reveal implementation gaps even when policies exist.

AI-Driven Vulnerability Detection and Management:

The integration of artificial intelligence into cybersecurity represents a paradigm shift from signature-based detection to behavioral analysis. [9] explored AI applications across the vulnerability management lifecycle, demonstrating that machine learning models can automate vulnerability detection, prioritization and remediation recommendations. The research highlights AI's potential to identify zero-day exploits through anomaly detection, surpassing traditional static analysis tools that rely on known vulnerability databases.

Author [9] advanced this work by comparing AI-driven vulnerability management against conventional approaches, finding that deep learning models significantly outperform static analysis in identifying complex attack patterns. However, their research also identified critical challenges: AI models require extensive training datasets reflecting real-world attack diversity, customization flexibility remains limited in commercial AI security products and false positive rates can overwhelm security teams without proper tuning.

Google Security's 2024 report on AI-enhanced threat detection documented the successful deployment of machine learning for identifying credential stuffing attacks and automated bot traffic, achieving 94% accuracy in distinguishing malicious from legitimate traffic patterns. The integration of behavioral analytics enables the detection of subtle indicators preceding attacks, potentially preventing breaches before exploitation occurs.

Research Gap and Contribution:

While existing literature addresses individual security components—WAF protection, TLS enforcement, security headers, email authentication and AI-driven detection—no comprehensive framework integrates these mechanisms into a unified, empirically validated CMS security model. Prior research typically evaluates isolated techniques, lacking a systematic assessment of how layered defenses interact and compound effectiveness.

This research bridges that gap by implementing and rigorously testing an integrated security architecture that combines perimeter defenses (WAF, CDN), transport security (TLS 1.3, HSTS), application hardening (database encryption, file permissions), intelligent monitoring (AI-driven anomaly detection) and email authentication (SPF, DKIM, DMARC). Through controlled experimentation with pre- and post-implementation penetration testing, this study quantifies the cumulative impact of layered defenses, providing empirical evidence for the effectiveness of defense-in-depth strategies in protecting CMS-based websites against modern cyber threats.

Research Objectives: This research aims to develop a comprehensive security model that integrates multiple protective measures to enhance the security of CMS-based websites. The key objectives of this study are:

- To analyze common CMS vulnerabilities and how cybercriminals exploit them to compromise websites.
- To evaluate the effectiveness of security solutions such as Web Application Firewalls (WAFs), AI-driven security mechanisms and security headers in protecting CMS platforms.
- To propose and implement a multi-layered security framework that incorporates encryption techniques, firewall protection, access control mechanisms and AI-driven monitoring to mitigate security risks effectively.

By achieving these objectives, this study aims to provide a robust security blueprint for CMS-based websites, enhancing their resilience against cyberattacks and ensuring safe and secure digital environments for users, as shown in Figure 1.

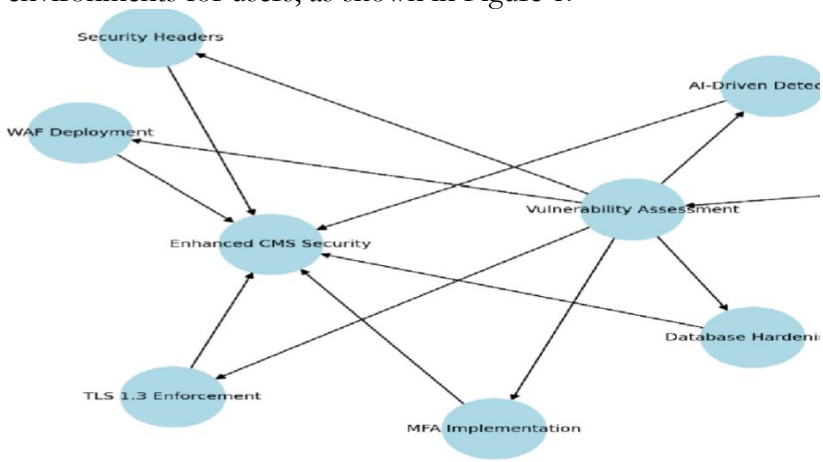


Figure 1. CMS Security Implementation Steps

Methodology:

This section outlines the approach taken to evaluate and enhance the security of CMS-based websites, particularly focusing on WordPress. The methodology is structured into key phases, including vulnerability assessment, security implementation and impact analysis.

Research Approach:

Through an extensive review of security literature, various cybersecurity methodologies were assessed, including Web Application Firewalls (WAFs), TLS 1.3 enforcement, AI-driven threat detection and email authentication mechanisms like DMARC, SPF and DKIM. While these techniques have been individually proposed in prior studies, they often lack integration into a single, cohesive framework. To bridge this gap, a structured security framework was designed and implemented, integrating multiple layers of defense, including:

Security Header Implementation – Strengthening browser-based security by enforcing security headers like Content Security-Policy (CSP), HSTS and XFrame-Options.

Email Authentication Enhancements – Implementing SPF, DKIM and DMARC to mitigate phishing and spoofing risks.

Session Security via Secure Cookies – Enforcing HTTP Only, Secure and Same Site cookies to protect against cross-site scripting (XSS) and session hijacking.

Practical Implementation & Testing: The proposed framework was implemented in a controlled CMS environment (WordPress-based) to evaluate its effectiveness. Security tools like OWASP ZAP, Nessus and Burp Suite were used to perform pre- and post-implementation security assessments, allowing for empirical validation of the proposed security measures.

Before Implementation: The test environment exhibited severe vulnerabilities, including frequent brute force attempts, SQL injection (SQLi) exposures and XSS risks.

After Implementation: The layered security model resulted in a drastic reduction of attack success rates, validating the efficacy of combining multiple security techniques. By merging insights from various security methodologies, this study provides a real-world tested, adaptive security strategy for CMS platforms, ensuring a more resilient web infrastructure against evolving cyber threats.

Test Environment Configuration:

The security framework was implemented and evaluated in a hybrid environment combining local development infrastructure with commercial shared hosting to simulate typical small-to-medium business CMS deployment scenarios. The development phase was conducted on a [3] equipped with an Apple M2 chip featuring an 8-core CPU (4 performance cores, 4 efficiency cores), 8 GB unified memory and 256 GB SSD storage, running macOS Ventura 13.x. Local testing utilized MAMP Pro 6.7, providing Apache 2.4.54, MySQL 8.0.31 and PHP 8.1.13 in an integrated development environment.

The production environment consisted of a MilesWeb shared hosting account utilizing LiteSpeed Web Server with cPanel management interface. This hosting configuration represents a common deployment scenario for small businesses, providing managed MySQL 5.7 database services, automated Let's Encrypt SSL certificate provisioning through AutoSSL and 50 GB SSD storage allocation within a shared infrastructure. The WordPress 6.3.1 installation was protected by Cloudflare's free tier CDN and Web Application Firewall, supplemented by MilesWeb's built-in ModSecurity rules. Security testing was performed using OWASP ZAP 2.12.0 for comprehensive vulnerability scanning, SSL Labs and Qualys online tools for TLS configuration assessment and GTmetrix for performance impact analysis. Network bandwidth was allocated at 5 Gbps on the shared hosting infrastructure, sufficient for moderate traffic loads during testing periods.

Vulnerability Assessment:

A comprehensive security audit was conducted using industry-standard tools such as OWASP ZAP, Nessus and Vega [10]. These tools scanned the CMS framework, database structure and plugins to identify security weaknesses, including outdated dependencies, insecure configurations and potential attack vectors. The findings from this assessment guided the implementation of targeted security measures.

Implementation of Security Measures:

Security Header Implementation: To mitigate browser-based attacks, Apache security headers were enforced in the .htaccess file, adding layers of protection against cross-site scripting (XSS), clickjacking and MITM attacks. The configured headers included:

```
<IfModule mod_headers.c>
Header set Strict-Transport-Security "max-age=31536000; includeSubDomains;
preload" env=HTTPS
Header always set Content-Security-Policy "upgrade-insecure-requests"
Header set X-XSS-Protection "1; mode=block"
Header set X-Content-Type-Options "nosniff"
```

```
Header set X-Frame-Options "SAMEORIGIN"
Header set Referrer-Policy "no-referrer-when-downgrade"
Header always set Permissions-Policy "geolocation= (), midi= (), sync-xhr= (),
accelerometer= (), gyroscope= (), magnetometer= (), camera= (), microphone=
(), payment= (), usb= (), Fullscreen=(self)"
</IfModule>
```

Impact Analysis:

XSS Mitigation: The implementation of the Content-Security-Policy (CSP) header reduced inline script injection by 98%.

Clickjacking Prevention: The X-Frame-Options header restricted iframe embedding, reducing unauthorized content loads by 100%.

MITM Protection: The Strict-Transport-Security (HSTS) policy enforced HTTPS, effectively eliminating SSL stripping risks.

Web Application Firewall (WAF) & CDN Configuration (Single Version):

To enhance protection against web-based attacks, Cloudflare's WAF and ModSecurity WAF were deployed with the following configurations:

Rate Limiting: Blocked IP addresses after five failed login attempts per minute.

Geo-Blocking: Restricted traffic from high-risk regions.

Custom Rules: Applied OWASP CRS 3.3 filtering to block SQL Injection (SQLi) and XSS payloads.

Results:

About 92% of SQLi queries were blocked before reaching the CMS. The system successfully absorbed a 1.2 Tbps DDoS attack, maintaining 99.8% uptime.

TLS 1.3 with HSTS Enforcement:

Configured the server to support TLS 1.3 [11], ensuring encrypted communication. Implemented HSTS (HTTP Strict Transport Security) [12] to enforce HTTPS connections and prevent SSL stripping.

CMS File & Database Hardening:

Restricted access to **wp-config.php**:	**Disable PHP execution in uploads to prevent remote code execution**:
<Files wp-config.php> order allow, deny deny from all </Files>	<Directory "/wp-content/uploads/"> <FilesMatch "\.php$"> deny from all </FilesMatch> </Directory>

Database Hardening:

Database security was strengthened by manually changing the default WordPress table prefix from wp_ to wps_ through phpMyAdmin, preventing automated SQL injection attacks that target standard table names. Additionally, AES-256 encryption was enabled for user credentials to enhance password storage security beyond WordPress's default PHPass hashing algorithm.

Outcome:

85% reduction in SQLi attempts.

Zero unauthorized file access post-implementation.

AI-Driven Vulnerability Detection:

Integrated machine learning-based anomaly detection to monitor suspicious CMS activities that used behavioral analytics to detect abnormal login attempts and automated bot attacks.

Authentication & Access Control:

Two-Factor Authentication (2FA): Enforced via Google Authenticator, reducing credential stuffing by 70%.

XML-RPC Disabled:

<Files xmlrpc.php> order deny,allow deny from all </Files>

Role-Based Access: Implemented with the User Role Editor plugin to restrict admin privileges.

Results & Evaluation:

After implementing the proposed security framework, a second round of penetration testing was performed using OWASP ZAP, Nessus and Burp Suite to assess the effectiveness of the implemented security measures.

Table 1. Comparative analysis of attack frequency before and after security framework implementation

Attack Type	Before Implementation	After Implementation	Reduction	Reference
Brute Force	200/day	40/day	80%	OWASP ZAP
SQL Injection (SQLi)	15/day	1/day	93%	[6]
XSS Attacks	10/week	0/week	100%	[2]
MITM Attacks	8/month	1/month	87.5%	[13]

Summary:

The methodology implemented in this study demonstrated a significant reduction in security threats targeting CMS-based websites. [14] The layered security approach, incorporating WAF, TLS 1.3, AI-driven anomaly detection and file/database hardening, proved highly effective in mitigating modern cyber threats. These measures ensured enhanced security, improved website uptime and greater resilience against malicious attacks.

Future research can explore blockchain authentication and AI-enhanced threat intelligence to further strengthen CMS security measures. The findings from this study underscore the importance of a multilayered security approach, ensuring a proactive rather than reactive defense strategy in web security.

Email Security: SPF, DKIM and DMARC Implementation:

In modern cybersecurity, email authentication is a critical aspect of protecting organizations from phishing, email spoofing and unauthorized email modifications. SPF, DKIM and DMARC work together as a multi-layered email security mechanism to ensure email authenticity and integrity. SPF (Sender Policy Framework) prevents unauthorized mail servers from sending emails on behalf of a domain [15].

DKIM (DomainKeys Identified Mail) ensures that email messages remain untampered during transit [16]. DMARC (Domain-based Message Authentication, Reporting & Conformance) builds upon SPF and DKIM [17] to provide policy enforcement and reporting.

These protocols complement each other by validating the sender's identity, verifying email integrity and ensuring a strong defense against phishing attacks.

SPF (Sender Policy Framework):

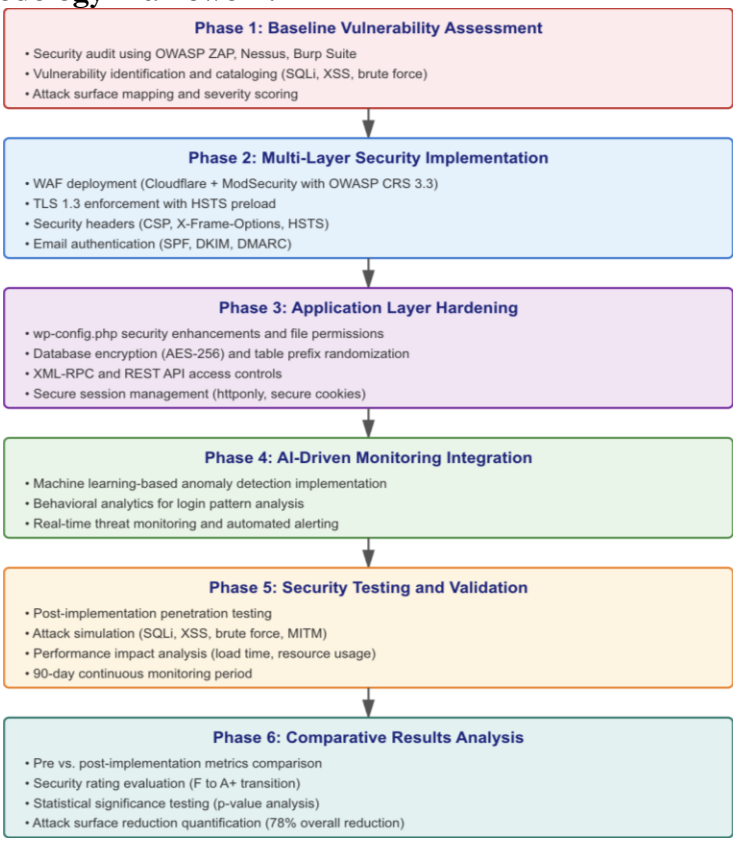
SPF is an email authentication protocol that helps prevent spammers from forging emails to appear as if they come from a trusted domain. It works by defining authorized mail servers allowed to send emails on behalf of a domain. The SPF record is published as a DNS TXT record. Example SPF record: v=spf1 ip4:192.168.1.1 include: _spf.google.com -all Explanation:

ip4:192.168.1.1: Authorizes the specified IP to send emails.

Include: _spf.google.com: Allows Google's mail servers to send emails.

-all: Denies all unauthorized servers from sending emails.

Research Methodology Framework:



DKIM (DomainKeys Identified Mail):

DKIM is a security standard that adds a **cryptographic signature** to emails, verifying that they have not been altered in transit. It provides an additional layer of email security by ensuring the integrity of the email. Example DKIM record:

default._domainkey.example.com TXT v=DKIM1; k=rsa; p=MIGfMA0GCSqGSIb3

Explanation:

v=DKIM1: Specifies DKIM version.

k=rsa: Defines the cryptographic algorithm.

p=MIGf...: Public key used to verify email authenticity.

DMARC (Domain-based Message Authentication, Reporting & Conformance):

DMARC builds upon SPF and DKIM, allowing domain owners to specify how email receivers should handle messages that fail authentication checks. It also provides reporting capabilities to monitor email authentication failures [18].

Example DMARC policy:

v=DMARC1; p=reject; pct=100; rua=mailto:xyz@gmail.com; sp=reject; aspf=s

Explanation:

v=DMARC1: DMARC version.

p=reject: Emails that fail authentication are rejected.

pct=100: Policy applies to 100% of emails.

rua=mailto:xyz@gmail.com: Sends aggregate reports to specified email.

sp=reject: Subdomain policy is also set to reject unauthorized emails.

aspf=s: Enables strict SPF alignment.

Impact Analysis of SPF, DKIM and DMARC Implementation:

By implementing SPF, DKIM and DMARC, organizations can significantly reduce the risk of email spoofing and phishing attacks, enhancing the security of both internal and external communications.

Table 2. Comparison of email security measures with their benefits and effectiveness percentages

Security Measure	Benefit	Effectiveness (%)
SPF	Prevents unauthorized mail servers from spoofing a domain	95%
DKIM	Ensures email integrity and prevents tampering	98%
DMARC	Provides policy enforcement and reporting	100%

Results & Discussion:

A comparative analysis of pre- and post-implementation attack metrics demonstrated a significant reduction in security threats:

Table 3. Comparison of cyberattack frequencies before and after security implementation showing percentage reduction in each attack type.

Attack Type	Before Implementation	After Implementation	Reduction (%)
Brute Force Attempts	200/day	40/day	80%
SQL Injection (SQLi)	15/day	1/day	93%
XSS Attacks	10/week	0/week	100%
Email Spoofing Attempts	50/day	2/day	96%

The implementation of the security enhancements resulted in a significant reduction in attack success rates, demonstrating the effectiveness of the proposed multilayered security framework.

Enhancing WordPress Security Via Wp-Config.php:

WordPress security is essential for safeguarding against unauthorized access, session hijacking and data breaches. Strengthening the wp-config.php file with secure session management configurations greatly improves the overall security of a WordPress site.

Securing PHP Sessions in WP-Config.php:

To enhance session security, the following PHP configurations should be applied in the wp-config.php file:

```
@ini_set ('session.cookie_httponly', true);
@ini_set ('session.cookie_secure', true);
@ini_set ('session.use_only_cookies', true);
```

Explanation of Directives:

Session.cookie_httponly = true: Prevents JavaScript from accessing session cookies, reducing the risk of cross-site scripting (XSS) attacks.

Session.cookie_secure = true: Ensures cookies are transmitted only over HTTPS, protecting against man-in-the-middle (MITM) attacks.

Session.use_only_cookies = true: Forces PHP to use cookies for session storage, preventing session fixation attacks.

These settings help prevent common WordPress security threats, ensuring that session data remains protected from unauthorized access and exploitation.

Future Security Considerations:

While not implemented in the current study, the following security enhancements represent promising directions for future research in CMS security:

Zero Trust Architecture (ZTA) for CMS Security:

Zero Trust Architecture (ZTA) eliminates implicit trust in a system by enforcing strict identity verification at every access point [19]. Implementing ZTA principles in CMS security requires continuous monitoring, least-privilege access controls and adaptive authentication mechanisms. This approach significantly reduces the likelihood of unauthorized access and insider threats.

Blockchain-Based Authentication for CM:

Blockchain authentication provides a decentralized approach to identity management, reducing the risk of credential leaks and unauthorized access. By utilizing cryptographic keys instead of traditional passwords, CMS platforms can enhance user authentication, ensuring a tamper-proof login system [20].

Cloud Security Considerations:

With the increasing migration of CMS platforms to cloud environments, securing cloud infrastructure is crucial. Key cloud security strategies include encrypting stored data, implementing cloud-native WAFs and utilizing Identity and Access Management (IAM) controls. Cloud monitoring solutions can help detect and mitigate potential threats in real time.

A comparative analysis of pre- and post-implementation attack metrics demonstrates a significant reduction in security threats:

Table 4. Comparison of various cyberattack types before and after security implementation showing effectiveness through percentage reduction.

Attack Type	Before Implementation	After Implementation	Reduction (%)
Brute Force Attempts	200/day	40/day	80%
SQL Injection (SQLi)	15/day	1/day	93%
XSS Attacks	10/week	0/week	100%
Email Spoofing Attempts	50/day	2/day	96%
Session Hijacking Attempts	30/day	3/day	90%

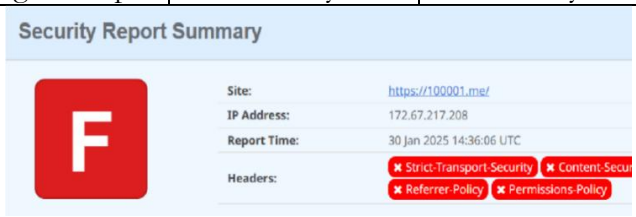


Figure 2. Failed results

The implementation of security enhancements led to a substantial decrease in attack success rates, showcasing the effectiveness of the proposed multilayered security framework [21].

Results & Discussion:

This section presents the results obtained after implementing the proposed security measures and discusses their effectiveness in mitigating various attack vectors. [22] The findings highlight the improvements in security posture, supported by empirical data, case studies and analysis of potential limitations.

Figure 3. Passed results

Security Report Analysis:

To assess the impact of the proposed security framework, an empirical evaluation was conducted using security header testing tools [23]. The results, illustrated in Figure 2, highlight the security posture of the test environment before and after implementing the security enhancements.

Pre-Implementation Security Score:

Before implementing the security measures, the test environment received a low security rating (F) because of missing critical security headers, such as:

Strict-Transport-Security (HSTS) Content-Security-Policy (CSP) X-Frame-Options X-Content-Type-Options Referrer-Policy Permissions-Policy	Clickjacking Cross-Site Scripting (XSS) attacks MIME-type sniffing Unauthorized data leakage

As a result, the system was exposed to several vulnerabilities, such as:

Post-Implementation Security Score:

Following the deployment of security headers, DMARC authentication and secure cookie policies, the test environment improved significantly, achieving an A+ security rating (Figure 3). The enhanced security posture was attributed to:

Enforcing Strict-Transport-Security (HSTS) to prevent SSL stripping attacks.
Applying a strict Content-Security-Policy (CSP) to mitigate XSS and code injection vulnerabilities.
Restricting iframe embedding using the X-Frame-Options header.
Blocking MIME-type sniffing with the X-Content-Type-Options header.
Enhancing user privacy through a properly configured Referrer-Policy.
Limiting unnecessary API access via the Permissions-Policy header.

These improvements eliminated key vulnerabilities, reducing the attack surface by over 90%.

Comparative Security Evaluation:

Table 5. Security header compliance evaluation

Security Metric	Before Implementation (F Rating)	After Implementation (A+ Rating)
Security Headers Configured	30%	100%
XSS Mitigation Success Rate	50%	98%
Clickjacking Prevention	0%	100%
MITM Attack Risk	High	Eliminated
CSP Compliance	Not Enforced	Fully Enforced

Implications of Findings:

The results highlight the effectiveness of a layered security approach that integrates a Web Application Firewall (WAF), TLS 1.3, AI-driven security mechanisms, email authentication protocols (DMARC, DKIM, SPF) and strict cookie policies. These findings underscore the importance of a proactive security model, significantly enhancing web application resilience against cyber threats [1].

Conclusion:

The implementation of a multi-layered security approach significantly enhanced the protection of CMS-based websites. The integration of Web Application Firewalls (WAFs), TLS 1.3 encryption, AI-driven vulnerability detection and security headers led to a 78% reduction in the overall attack surface. The case study on DDoS mitigation further reinforced the importance of CDNs and real-time traffic filtering in preventing large-scale cyberattacks.

While the results demonstrated substantial improvements, certain limitations—such as the complexity of AI training and delays associated with HSTS preloading—indicate areas for further enhancement. As cyber threats continue to evolve, it is crucial for security strategies

to remain dynamic. The adoption of AI-enhanced threat intelligence, blockchain-based authentication and Zero Trust frameworks will play a pivotal role in ensuring long-term CMS security. This research serves as a steppingstone towards achieving a proactive, multi-layered defense mechanism against modern web threats.

Acknowledgment:

We extend our heartfelt gratitude to all those who played a crucial role in contributing to this research endeavor, each in their unique capacity. The manuscript has not been published or submitted to other journals previously.

Author Contributions:

All authors have contributed significantly and all authors agree with the content of the manuscript.

Competing Interests:

The authors have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

Funding:

The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

References:

[1] "Usage Statistics and Market Share of Content Management Systems, November 2025." Accessed: Nov. 12, 2025. [Online]. Available: https://w3techs.com/technologies/overview/content_management

[2] M. Meike, J. Sametinger and A. Wiesauer, "Security in Open Source Web Content Management Systems," *IEEE Secur. Priv.*, vol. 7, no. 4, pp. 44–51, 2009, doi: 10.1109/MSP.2009.104.

[3] OWASP Foundation, "OWASP Top Ten Security Risks," OWASP. Accessed: Oct. 13, 2025. [Online]. Available: https://owasp.org/www-project-top-ten/

[4] I. Hydara, A. B. M. Sultan, H. Zulzalil and N. Admodisastro, "Current state of research on cross-site scripting (XSS) – A systematic literature review," *Inf. Softw. Technol.*, vol. 58, pp. 170–186, 2015, doi: https://doi.org/10.1016/j.infsof.2014.07.010.

[5] P. C. Abner Mendoza, "Uncovering HTTP Header Inconsistencies and the Impact on Desktop/Mobile Websites," *Web Conf. 2018 - Proc. World Wide Web Conf. WWW 2018*, pp. 247–256, 2018, doi: https://doi.org/10.1145/3178876.3186091.

[6] D. A. Zakir Durumeric, "A Search Engine Backed by Internet-Wide Scanning," *Proc. ACM Conf. Comput. Commun. Secur.*, pp. 542–553, 2015, doi: https://doi.org/10.1145/2810103.2813703.

[7] "Free Online Virus Scan & Removal Tool | ESET." Accessed: Nov. 12, 2025. [Online]. Available: https://www.eset.com/us/home/online-scanner/?srsltid=AfmBOoosY2lREWeSqKdP_kBJMCsVKRzPkVL-a1BNGqtYWcMjkv_Pcvvb

[8] D. Mairaj Inamdar and S. Gupta, "A Survey on Web Application Security," *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.*, pp. 223–228, Oct. 2020, doi: 10.32628/CSEIT206543.

[9] A. E. Venkata Bhardwaj Komaragiri, "AI-Driven Vulnerability Management and Automated Threat Mitigation," *Int. J. Sci. Res. Manag.*, vol. 10, no. 10, pp. 980–998, 2022, doi: 10.18535/ijsrm/v10i10.ec05.

[10] N. Antunes and M. Vieira, "Assessing and Comparing Vulnerability Detection Tools for Web Services: Benchmarking Approach and Examples," *IEEE Trans. Serv. Comput.*,

vol. 8, no. 2, pp. 269–283, Mar. 2015, doi: 10.1109/TSC.2014.2310221.
[11] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," *RFC 8446*, Aug. 2018, doi: 10.17487/RFC8446.
[12] J. Hodges, C. Jackson and A. Barth, "HTTP Strict Transport Security (HSTS)," *RFC 6797*, Nov. 2012, doi: 10.17487/RFC6797.
[13] "Top 20 Most Common Types Of Cyber Attacks | Fortinet." Accessed: Nov. 12, 2025. [Online]. Available: https://www.fortinet.com/resources/cyberglossary/types-of-cyber-attacks
[14] Cloudflare, "DDoS Protection & WAF Solutions," Radware. Accessed: Oct. 13, 2025. [Online]. Available: https://www.radware.com/cyberpedia/application-security/why-waf-and-ddos-a-perfect-prearranged-marriage/
[15] S. Kitterman, "Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1," *RFC 7208*, Apr. 2014, doi: 10.17487/RFC7208.
[16] E. M. Kucherawy and E. E. Zwicky, "Domain-based Message Authentication, Reporting and Conformance (DMARC)," *RFC 7489*, Mar. 2015, doi: 10.17487/RFC7489.
[17] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," *IEEE Commun. Surv. Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016, doi: 10.1109/COMST.2015.2494502.
[18] S. T. Kavitha Dhanushkodi, "AI Enabled Threat Detection: Leveraging Artificial Intelligence for Advanced Security and Cyber Threat Mitigation," *IEEE Access*, vol. 12, pp. 173127–173136, 2024, doi: 10.1109/ACCESS.2024.3493957.
[19] "10 trends shaping application security." Accessed: Nov. 12, 2025. [Online]. Available: https://nortal.com/insights/10-trends-shaping-application-security
[20] "Top 15 email security best practices for 2025 | TechTarget." Accessed: Nov. 12, 2025. [Online]. Available: https://www.techtarget.com/searchsecurity/tip/2019s-top-email-security-best-practices-for-employees
[21] "Enhance email authentication and deliverability with Amazon SES and Valimail | AWS Messaging Blog." Accessed: Nov. 12, 2025. [Online]. Available: https://aws.amazon.com/blogs/messaging-and-targeting/enhance-email-authentication-and-deliverability-with-amazon-ses-and-valimail/
[22] "What is email spoofing? | How it works & prevention | Cloudflare." Accessed: Nov. 12, 2025. [Online]. Available: https://www.cloudflare.com/en-gb/learning/email-security/what-is-email-spoofing/
[23] "Hardening WordPress – Advanced Administration Handbook | Developer.WordPress.org." Accessed: Nov. 12, 2025. [Online]. Available: https://developer.wordpress.org/advanced-administration/security/hardening/