Journal of Information Systems and Informatics

Vol. 7, No. 3, September 2025 e-ISSN: **2656-4882** p-ISSN: **2656-5935**

DOI: 10.51519/journalisi.v7i3.1218 Published By DRPM-UBD

Enhancing Web Application Security with Open-AppSec WAF on CDN Infrastructure

Andi Yusdar Al Imran[1], Muhammad Nur Yasir Utomo[2], Iin Karmila Yusri[3]

[1,2,3]Department of Informatics and Computer Engineering, State Polythecnic of Ujung Pandang, Makassar, Indonesia

Email: [1]andiyusdaralimran@gmail.com, [2]yasirutomo@poliupg.ac.id, [3]iin.yusri@poliupg.ac.id

Abstract

The increasing number of cyberattacks targeting web applications has made security a critical concern, with vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), Broken Authentication, and Cross-Site Request Forgery (CSRF) remaining prevalent in the OWASP Top 10. These threats can lead to data breaches, service disruption, and reputational damage if not properly mitigated. To address this issue, an infrastructure combining Open-AppSec Web Application Firewall (WAF) and Varnish Cache Content Delivery Network (CDN) was implemented on a Moodle-based e-learning platform within a virtualized Proxmox VE environment. Security testing was conducted using OWASP ZAP and Burp Suite under two scenarios: without WAF and with WAF. In the first scenario, OWASP ZAP detected multiple vulnerabilities, and Burp Suite confirmed successful exploitation with 200 OK responses. In the second scenario, all vulnerabilities were eliminated, and all simulated attacks returned 403 Forbidden responses, indicating complete mitigation. Performance tests revealed a manageable overhead, with throughput reaching 115.4 req/sec at 1000 concurrent users, accompanied by a slight increase in response time and latency. These results demonstrate that integrating Open-AppSec with CDN infrastructure can effectively protect against application-layer attacks while maintaining optimal content delivery performance. Limitations of this study include testing within a simulated environment; therefore, future work could validate these findings on larger-scale systems and with real-world traffic to assess broader generalizability.

Keywords: Open-AppSec, Web Application Firewall, CDN, Web Security, OWASP, Machine Learning

1. INTRODUCTION

The evolution of web technology has fundamentally reshaped how information and services are accessed in modern society. From online shopping and banking to education and public administration, web applications are now integral to everyday digital interactions [1]. In the context of education, e-learning platforms have become critical, particularly following the global transition to remote learning after the COVID-19 pandemic. These platforms, such as Moodle, serve thousands of concurrent users and handle sensitive data ranging from user credentials to examination results [2]. However, the increasing dependence on web applications has also widened the attack surface for cyber threats [3].

As the volume of traffic increases, ensuring optimal performance and robust security becomes a challenging task. Content Delivery Networks (CDNs) offer a practical solution to scalability and performance issues by distributing cached content across geographically dispersed servers [4], [5]. This reduces latency and alleviates the load on the origin server, improving responsiveness even during high-traffic events such as online exams [6], [7]. However, despite performance improvements, CDN infrastructure alone does not provide adequate security [8]. Malicious traffic can still reach the edge servers and compromise the origin server if not filtered properly [9].

According to a survey by Pingale & Sutar, over 72% of hosted websites are vulnerable to web-based attacks, including SQL Injection and Cross-Site Scripting (XSS) [3]. These attacks exploit weaknesses in input validation and session management, leading to data leakage, privilege escalation, and even full system compromise. Another growing concern is the emergence of zero-day attacks—threats that exploit unknown vulnerabilities before a patch becomes available. Traditional Web Application Firewalls (WAFs), such as ModSecurity, rely on static rule sets that must be updated manually [10]. This results in limited adaptability and high false positive rates, making them ineffective against modern threat vectors [11].

To address these challenges, a paradigm shift towards intelligent, adaptive security solutions is needed. Open-AppSec is an open-source WAF developed by Check Point that leverages artificial intelligence and machine learning to detect and prevent application-layer attacks in real time [12]. Unlike signature-based WAFs, Open-AppSec can learn from traffic patterns and adjust its detection heuristics automatically [13]. It supports integration with modern server stacks such as NGINX, Kubernetes, and cloud-native platforms, making it highly versatile [14].

Despite these advancements, most existing studies have focused either on CDN performance optimization [4], [6], [15] or the use of traditional WAFs in isolation [10], [11]. Research specifically examining the integration of Open-AppSec with CDN infrastructures remains limited, leaving a gap in understanding how this combination can simultaneously enhance web application security and maintain system performance. Moreover, while Open-AppSec provides adaptive protection compared to static-rule WAFs, its deployment also introduces potential challenges such as resource consumption overhead and the risk of false positives [10], [11]. These factors underline the importance of empirical studies to evaluate both the benefits and trade-offs of applying Open-AppSec in CDN-based environments.

This research aims to investigate the effectiveness of Open-AppSec in enhancing the security of web applications deployed within a CDN architecture [16]. Specifically, the study implements a layered security model consisting of Moodle as the application layer, Varnish Cache as the CDN layer, and Open-AppSec as the security layer [17]. The deployment was conducted in a virtualized environment using Proxmox VE [18] and Docker containers to simulate real-world infrastructure [19], [20].

The research focuses on two main objectives: to implement Open-AppSec as a WAF on a CDN-based web architecture, and to evaluate the system's effectiveness in preventing OWASP Top 10 attacks, including SQL Injection, XSS, Broken Authentication, and Anti-CSRF. By comparing the system's behavior before and after WAF deployment [21] using OWASP ZAP and Burp Suite for black-box testing, the study aims to provide empirical evidence on the feasibility of AI-based WAFs for modern web environments.

2. METHODS

This section presents the research methodology used to implement and evaluate the security enhancement of web applications through the integration of Open-AppSec WAF into a CDN-based server environment. The methodology is divided into four stages: infrastructure design, system implementation, attack simulation, and effectiveness evaluation.

2.1 System Design

The system design stage is carried out to plan the infrastructure and flow of the CDN system:

2.1.1 Infrastructure

In this stage, the infrastructure design is carried out to ensure that the Web Application Firewall (WAF) and Content Delivery Network (CDN) system can function optimally in securing and accelerating the delivery of e-learning content. The designed infrastructure includes key elements such as clients, the WAF server with NGINX Proxy Manager and Open-AppSec, the CDN server with caching capability, the origin server hosting the Moodle-based e-learning platform, and the e-learning database.

Figure 1 illustrates the data flow from the client request to the response provided by the e-learning system. All incoming requests first pass through the WAF layer, which inspects and filters malicious traffic [18][19] before forwarding legitimate requests to the CDN server. The CDN serves frequently accessed static content directly from its cache[6], reducing latency and minimizing load on the origin server[14]. For dynamic requests, the CDN forwards the traffic to the origin server, which processes the request and interacts with the database to retrieve or store academic data [13]. By using this layered approach, the system ensures improved security [20], faster response times, reduced server workload, and greater stability, particularly for serving large-scale concurrent user access [21].

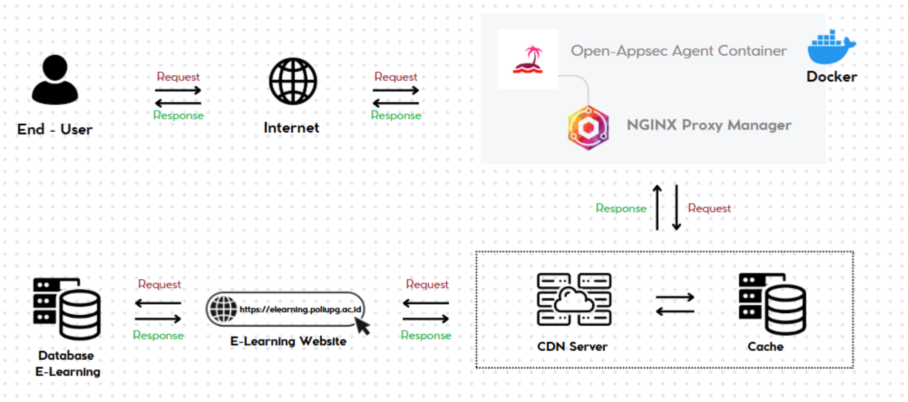

Figure 1. Open-Appsec with CDN Infrastructure Design

2.1.2 WAF - CDN Flow

Figure 2 illustrates the process that begins when a client sends a request to access the e-learning platform. Upon receiving the request, the WAF examines the traffic for potential security threats such as SQL injection, cross-site scripting, or malicious bots [18]. If the request is deemed malicious, it is blocked, and no further processing occurs [1]. If the request passes inspection, it is forwarded to the CDN server for content delivery. The WAF–CDN flow secures and optimizes content delivery by placing the WAF as the first entry point, filtering malicious traffic before it reaches the CDN.

At the CDN stage, the system checks whether the requested content is available in the cache. If the content exists (cache hit), the CDN serves it directly to the client, significantly reducing latency and server load [13]. Conversely, if the content is not cached (cache miss), the CDN forwards the request to the origin server, retrieves the content, and stores it in the cache for future requests. By combining WAF security inspection with CDN caching capabilities, this flow ensures that only legitimate traffic reaches the application while optimizing content distribution, accelerating access times, and enhancing the overall stability [20] of the e-learning system.

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

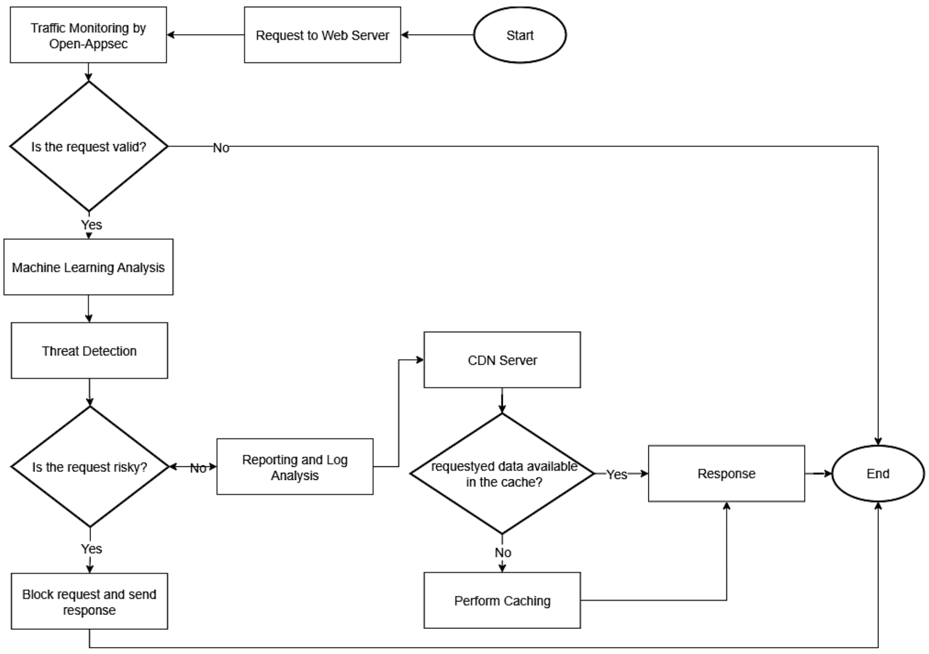
Start
Request to Web Server
Traffic Monitoring by Open-Appsec
Is the request valid?
No
Yes
Machine Learning Analysis
Threat Detection
Is the request risky?
No
Yes
Block request and send response
Reporting and Log Analysis
CDN Server
requestyed data available in the cache?
Yes
No
Perform Caching
Response
End

Figure 2. WAF - CDN Flow Design

2.2 Implementation

The system creation was carried out in Politeknik Negeri Ujung Pandang using Proxmox Virtual Environment (Proxmox VE) as the virtualization platform. The servers were provisioned with a 4-core CPU and 8 GB of RAM, specifications that align with the infrastructure requirements of the e-learning platform and are capable of supporting a large number of concurrent users, as shown in Table 1.

Table 1. Server Specification

Platform	Server	Specification
Proxmox	Origin Server (E-learning)	OS: Ubuntu Server 22.04 CPU: 4 Core RAM: 8 GB Disk: 25 GB
	CDN Server	OS: Ubuntu Server 22.04 CPU: 4 Core RAM: 8 GB Disk: 25 GB
	WAF Server	OS: Ubuntu Server 22.04

Platform	Server	Specification
		CPU: 4 Core RAM: 8 GB Disk: 25 GB

The origin server is configured with the Moodle Learning Management System (LMS), which is the official platform used by Politeknik Negeri Ujung Pandang. Moodle was selected due to its open-source nature, flexibility, and modular architecture, which supports integration with third-party tools and plugins, allowing it to accommodate a wide range of educational needs [2]. The CDN server employs Varnish Cache to store and deliver static content such as images, JavaScript, and CSS files [14]. This reduces latency, minimizes load on the origin server, and improves the scalability of the system during peak usage periods. The WAF server runs Open-AppSec integrated with NGINX Proxy Manager in a Docker container. This component acts as the first security layer, filtering incoming requests using machine learning-based threat detection, bot prevention, and custom security rules [22] before allowing them to proceed to the CDN or origin server.

2.2.1 Server Configuration

Server configuration includes as follow.

1) Installation of Ubuntu Server 22.04 LTS: All servers (Origin, CDN, and WAF) are installed with Ubuntu Server 22.04 LTS due to its stability, security, and compatibility with the required software components [23].
2) Installation of Apache Web Server: Apache is installed on the Origin Server to handle HTTP requests that are not served by the CDN cache. Apache processes dynamic requests such as quizzes, assignments, and user authentication [24] in Moodle.
3) Installation of Moodle LMS: Moodle is deployed on the Origin Server as the main e-learning platform. Its modular architecture supports a variety of plugins and third-party integrations for educational purposes [2].
4) Installation of MySQL Database Server: MySQL is configured on the Origin Server to store Moodle's database, including course materials, user information, and activity logs.

2.2.2 WAF - CDN Configuration

WAF–CDN configuration includes as follow.

1) Open-AppSec Installation: Open-AppSec is deployed inside a Docker container on the WAF server. It is integrated with NGINX Proxy Manager to inspect and filter incoming traffic, blocking potential threats [22] such as

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

SQL injection, cross-site scripting (XSS), and bot traffic before they reach the CDN or Origin Server.

2) Varnish Cache Installation: The CDN server is installed with Varnish Cache to store and serve static content, reducing latency and load [14] on the Origin Server.
3) Backend Configuration: Varnish is configured to route cache misses to the Origin Server (Nginx) as the backend, ensuring that dynamic content is served accurately [14].

2.3 Evaluation

After the WAF–CDN system is fully deployed, testing is carried out to evaluate its effectiveness in securing the e-learning platform. The evaluation compares system behavior in two scenarios: before the implementation of the WAF, where the web application is only served through the CDN, and after the implementation of the WAF, where all incoming traffic is filtered by Open-AppSec before reaching the CDN or origin server. The security tests were performed using two widely recognized penetration testing tools: OWASP ZAP and Burp Suite. OWASP ZAP is used to identify web application vulnerabilities by detecting the number and types of weaknesses present[25], while Burp Suite focuses on analyzing HTTP response codes generated during attack simulations [26]. The parameters measured in this evaluation include:

1) Number of Vulnerabilities Detected (ZAP): The total count of vulnerabilities identified in the application, classified by severity level (High, Medium, Low, Informational).
2) HTTP Response Codes (Burp Suite): The classification of server responses to simulated attacks, including:
   a) 200 OK – attack successful (no protection),
   b) 403 Forbidden – attack blocked (successful mitigation),
   c) 502 Bad Gateway – request failed due to misconfiguration or server error. [27]
3) Detection Rate: The percentage of attack attempts successfully blocked by the WAF.

By comparing the results from both tools before and after WAF deployment, the impact of Open-AppSec on application-layer security can be quantitatively assessed. The implementation is expected to significantly reduce the number of vulnerabilities, increase the proportion of blocked attacks (403 responses), and improve the overall security posture [19] of the e-learning platform.

2.4 Ethical Considerations

All security and performance testing activities were conducted with the full knowledge and approval of the IT management at Politeknik Negeri Ujung Pandang. To prevent any disruption to live educational activities and to protect the privacy of students and faculty, all evaluations were carried out within a controlled, sandboxed environment hosted on Proxmox VE. This environment was an exact replica of the production infrastructure but was completely isolated from the live network and contained no real user data. All tests, including vulnerability scanning and attack simulations, were performed using dummy accounts and sample course materials to ensure the integrity and confidentiality of academic records.

3. RESULT AND DISCUSSION

3.1 Evaluation by OWASP ZAP

The vulnerability assessment using OWASP ZAP was conducted to identify and classify security issues within the Moodle-based e-learning platform. The evaluation measured the number of findings across different severity levels before and after the deployment of the WAF, as shown in Table 2. These results indicate that Open-AppSec successfully mitigated all vulnerabilities detected in the Without WAF scenario, achieving a 100% mitigation rate across all tested attack types.

Table 2. OWASP ZAP Vulnerability Scan Results

Attack Type	Vulnerabilities Detected (No WAF)	Vulnerabilities Detected (With WAF)	Mitigation Rate (%)
SQL Injection	7	0	100
Cross-Site Scripting (XSS)	0	0	100
Broken Authentication	4	0	100
Cross-Site Request Forgery	0	0	100

3.2 Evaluation by Burp Suite

Burp Suite was employed to simulate various attack scenarios and monitor the HTTP response codes returned by the application. These codes indicate whether the attack was successful, blocked, or resulted in an error, as shown in Table 3. The 200 OK status in the Without WAF scenario confirms that the application was vulnerable to all tested attacks. In contrast, the 403 Forbidden statuses in the

With WAF scenario demonstrates that Open-AppSec blocked the malicious requests before they reached the application.

Table 3. Burp Suite Response Codes for Attack Attempts

Attack Type	HTTP Code (No WAF)	HTTP Code (With WAF)	Interpretation
SQL Injection	200 OK	403 Forbidden	Blocked successfully
Cross-Site Scripting (XSS)	200 OK	403 Forbidden	Blocked successfully
Broken Authentication	200 OK	403 Forbidden	Blocked successfully
Cross-Site Request Forgery	200 OK	403 Forbidden	Blocked successfully

3.3 Evaluation of Throughput

Table 4 presents the throughput results measured in requests per second (req/sec). As the number of concurrent users increases, throughput generally rises in both scenarios. Without WAF, throughput ranges from 47.3 req/sec at 10 users to 145.9 req/sec at 1000 users. With Open-AppSec enabled, throughput is slightly lower, ranging from 42.1 req/sec to 115.4 req/sec.

Table 4. Evaluation of Throughput

Users	Throughput (req/sec)	
	Without WAF	With WAF
10	47.3	42.1
50	57.6	51.3
100	83.5	73.5
200	87.6	76.7
300	101.5	87.7
500	122.9	102.8
700	137.2	111.4
1000	145.9	115.4

This reduction is expected due to the additional inspection and filtering performed by the WAF. Nevertheless, the system still demonstrates the ability to process a significant volume of requests, which shows that the performance overhead remains manageable. Figure 3 illustrates the same trend graphically, where the curve for "With WAF" consistently remains below "Without WAF," but both exhibit similar growth patterns as the number of users increases.

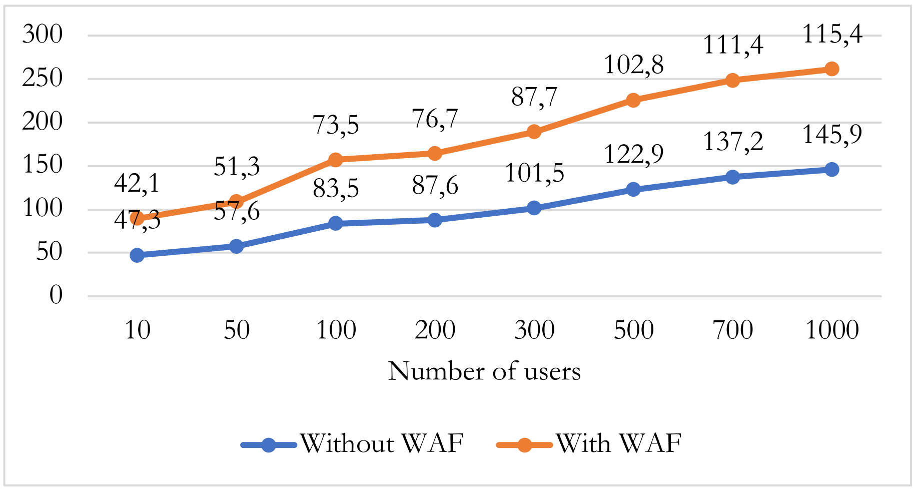

Figure 3. Throughput Comparison Chart

3.4 Evaluation of Response Time

Table 5 shows the evaluation of response time in milliseconds (ms). The results indicate a steady increase in response time as the number of users grows. At 10 users, the response time is 135 ms without WAF and 150 ms with WAF. At 1000 users, it reaches 7911 ms without WAF and 8238 ms with WAF.

Table 5. Evaluation of Response Time

Users	Response Time (ms)	
	Without WAF	With WAF
10	135	150
50	746	822
100	993	1021
200	2080	2114
300	3922	4076
500	4964	5155
700	6033	6296
1000	7911	8238

The addition of the WAF introduces only a modest increase in response time, which indicates that the overhead from Open-AppSec is relatively small compared to the overall processing time. Figure 3 provides a graphical representation, clearly showing the rising response time with higher user loads, with the WAF scenario slightly above the baseline but still maintaining a similar progression.

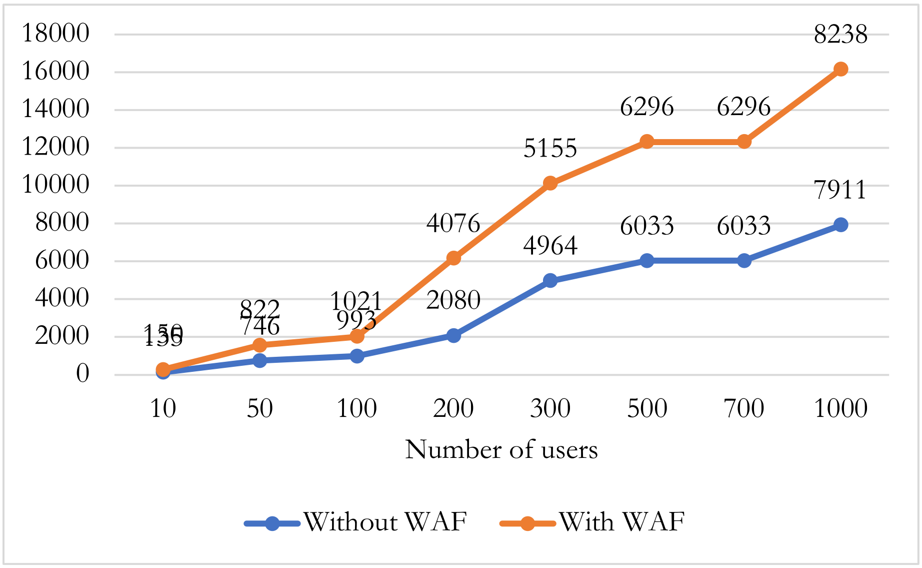

Figure 4. Response Time Comparison Chart

3.5 Evaluation of Latency

Table 6 presents the latency evaluation results. Similar to response time, latency increases with the number of concurrent users. For instance, latency at 10 users is 208.2 ms without WAF and 215.6 ms with WAF. At 1000 users, it rises to 3822.8 ms and 3921.3 ms, respectively.

Table 6. Evaluation of Latency

Users	Latency (ms)	
	Without WAF	With WAF
10	208.2	215.6
50	530.0	534.2
100	968.3	992.2
200	1160.9	1175.2
300	1592.2	1611.6

Users	**Latency (ms)**	
	Without WAF	**With WAF**
500	2261.5	2262.0
700	3725.1	3800.0
1000	3822.8	3921.3

The differences between both scenarios are relatively minor, ranging between 7–100 ms, which indicates that the WAF introduces only minimal delays. As shown in Figure 5, the latency trend lines for both conditions are nearly parallel, confirming that while Open-AppSec adds a small inspection overhead, overall system responsiveness remains acceptable

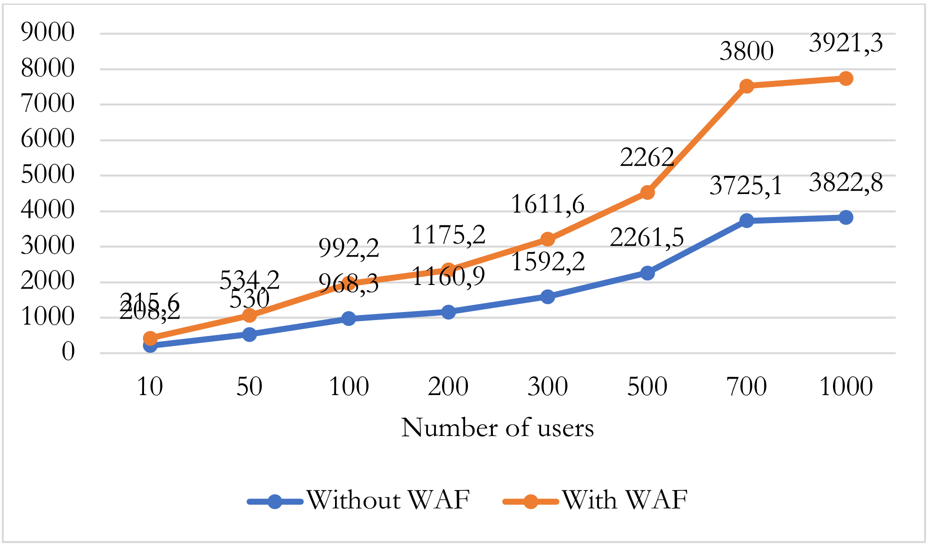

Figure 5. Latency Comparison Chart

3.6 Comparative Analysis

The difference in reporting between OWASP ZAP and Burp Suite highlights the complementary nature of these tools. OWASP ZAP excels in identifying and categorizing vulnerabilities, providing a quantitative measure of exposed attack surfaces. Burp Suite, on the other hand, is effective in validating whether the attack payload can bypass defenses, offering direct insight into request/response behavior. Combining the results from both tools allows for a more comprehensive evaluation of WAF effectiveness. In this study, Open-AppSec demonstrated complete mitigation across all tested vectors, confirming its capability to protect the application layer against common OWASP Top 10 threats.

Beyond security testing, the performance evaluation results further support this conclusion. While enabling Open-AppSec introduced a measurable performance overhead—throughput decreased by approximately 10–20% and both response time and latency increased slightly—the overall impact remained within acceptable operational limits. This trade-off is consistent with expectations, as packet inspection and traffic filtering inherently require additional processing resources. Importantly, the system remained stable under increasing user loads, and no abnormal degradation was observed. Taken together, the security validation through OWASP ZAP and Burp Suite, combined with the performance benchmarking, confirms that Open-AppSec provides a balanced approach: significantly enhanced security with only minor performance penalties. This makes Open-AppSec a practical and effective WAF solution for securing web applications deployed in CDN environments.

3.7 Discussion

The implementation of Open-AppSec as a WAF demonstrated significant improvements in the security posture of the Moodle-based e-learning platform. Vulnerability assessments using OWASP ZAP revealed multiple critical attack vectors in the baseline configuration, including SQL Injection and Broken Authentication, all of which were fully mitigated [19] once the WAF was deployed. Burp Suite confirmed these results by showing consistent blocking behavior, where all malicious requests returned 403 Forbidden responses. These outcomes highlight Open-AppSec's effectiveness in addressing OWASP Top 10 vulnerabilities.

The observed performance trade-offs can be explained by the additional inspection required for each HTTP/HTTPS request. Throughput decreased by 10–20% due to the overhead of traffic filtering, while response time and latency showed marginal increases, generally below 5%. This aligns with prior studies on WAF deployment [3], where similar overheads were reported. Importantly, the increases remained within acceptable thresholds, indicating that the performance penalty does not critically affect user experience.

Compared to traditional WAFs such as ModSecurity, which rely on static rules that are often prone to high false positive rates [28] and less effective against zero-day attacks, Open-AppSec offers adaptive, AI-driven detection mechanisms [18]. This capability not only explains its ability to block all tested attacks, including SQL Injection and XSS, without manual rule updates, but also aims to minimize false positives by learning normal traffic patterns. Within the scope of this study, which focused on clear-cut attack vectors, no significant false positives were observed.The results therefore confirm the practicality of Open-AppSec as a modern WAF that balances strong security, high accuracy, and efficient performance.

In an operational context, securing e-learning platforms [2] is essential given their role in handling sensitive academic data, such as student credentials, grades, and instructional content. Integrating Open-AppSec with a CDN not only ensures optimized content delivery but also safeguards against malicious traffic, maintaining service continuity even during attack attempts. This layered defense model strengthens institutional resilience against evolving cyber threats. Nevertheless, the study is not without limitations. The evaluation was conducted on a Moodle-based e-learning system within a simulated environment [15], with a maximum load of 1000 concurrent users. Future work could expand testing to larger-scale environments, real-world traffic datasets, or other types of web applications to further validate scalability and generalizability.

4. CONCLUSION

The implementation of Open-AppSec as a Web Application Firewall (WAF) in conjunction with a CDN resulted in a 100% mitigation rate for all vulnerabilities detected by OWASP ZAP, including SQL Injection, Cross-Site Scripting (XSS), Broken Authentication, and Cross-Site Request Forgery (CSRF). Similarly, Burp Suite simulations showed a 100% blocking rate, with all malicious requests returning 403 Forbidden responses after WAF deployment. This configuration ensures that only legitimate traffic reaches the CDN and origin server, safeguarding sensitive academic data and maintaining uninterrupted service for users. These results demonstrate that a properly configured WAF–CDN setup provides comprehensive protection against common OWASP Top 10 threats, making it an essential component for securing modern digital learning environments.

REFERENCES

[1] R. Riska and H. Alamsyah, "Penerapan Sistem Keamanan Web Menggunakan Metode Web Aplication Firewall," *J. Amplif. J. Ilm. Bid. Tek. ELEKTRO DAN Komput.*, vol. 11, no. 1, 2021, doi: 10.33369/jamplifier.v11i1.16683.

[2] R. Irfan and C. Y. Pratama, "Improvement of Performance E-Learning Moodle Service in Vocational High School with Optimization of Web Server and Database Server," *Elinvo (Electronics, Informatics, Vocat. Educ.*, vol. 9, no. 1, pp. 52–63, 2024, doi: 10.21831/elinvo.v9i1.42878.

[3] S. V. Pingale and S. R. Sutar, "Analysis of Web Application Firewalls, Challenges, and Research Opportunities," *Lect. Notes Electr. Eng.*, vol. 783, no. January 2022, pp. 239–248, 2022, doi: 10.1007/978-981-16-3690-5_21.

[4] L. Gao and X. Zhu, "ICN-Based Enhanced Content Delivery for CDN,"

Futur. Internet, vol. 15, no. 12, 2023, doi: 10.3390/fi15120390.
[5] A. Ghasemi and A. Ahmadi, "Cache management in content delivery networks using the metadata of online social networks," *Comput. Commun.*, vol. 189, pp. 11–17, 2022, doi: 10.1016/j.comcom.2022.02.021.
[6] A. H. Ibrahim, Z. T. Fayed, and H. M. Faheem, "Fog-based CDN framework for minimizing latency of web services using fog-based HTTP browser," *Futur. Internet*, vol. 13, no. 12, 2021, doi: 10.3390/fi13120320.
[7] V. Sathiyamoorthi, P. Suresh, N. Jayapandian, P. Kanmani, M. Deva Priya, and S. Janakiraman, "An intelligent web caching system for improving the performance of a web-based information retrieval system," *Int. J. Semant. Web Inf. Syst.*, vol. 16, no. 4, 2020, doi: 10.4018/IJSWIS.2020100102.
[8] D. Laksmiati, "Implementasi Content Delivery Network (CDN) untuk optimasi kecepatan akses website," *Akrab Juara: Jurnal Ilmu-ilmu Sosial*, vol. 5, no. 1, pp. 49-56, 2020.
[9] Z. Li and W. Meng, "Mind the Amplification: Cracking Content Delivery Networks via DDoS Attacks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2021. doi: 10.1007/978-3-030-86130-8_15.
[10] R. A. Muzaki, O. C. Briliyant, M. A. Hasditama, and H. Ritchi, "Improving Security of Web-Based Application Using ModSecurity and Reverse Proxy in Web Application Firewall," *2020 Int. Work. Big Data Inf. Secur. IWBIS 2020*, no. December, pp. 85–90, 2020, doi: 10.1109/IWBIS50925.2020.9255601.
[11] I. D. Wiradyaksa, D. H. Putri, R. M. Iqbal, N. H. Astari, N. Karna, and F. Dewanta, "Design and Implementation of Automated Web Application Firewall, Rate Limiting, and Intrusion Detection System for Cyber Defense," in *2024 8th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, Aug. 2024, pp. 256-261.
[12] F. Vrbić, "Analiza mogućnosti različitih API vatrozida i testiranje njihove primjene za PSD2," Ph.D. dissertation, University of Zagreb, Faculty of Electrical Engineering and Computing, 2024.
[13] D. Laksmiati, "Implementasi Content Delivery Network (Cdn) Untuk Optimasi Kecepatan Akses Website," *Akrab Juara*, vol. 5, no. 1, 2020.
[14] M. N. Y. Utomo, E. Tungadi, and W. Khartika, "Enhancing web performance for e-learning platform using content delivery network (CDN) and varnish cache," *Journal of Information Systems and Informatics*, vol. 7, no. 1, pp. 831-847, 2025, doi: 10.51519/journalisi.v7i1.993.
[15] R. Chandra and A. T. Sitorus, "Virtualisasi Server menggunakan Proxmox untuk mengoptimalkan Resource Server pada SMK Bhakti Persada," *Jurnal Multidisiplin Ilmu Akademik*, vol. 1, no. 2, pp. 69-80, 2024.
[16] S. Dwiyatno, E. Rachmat, A. P. Sari, and O. Gustiawan, "Implementasi Virtualisasi Server Berbasis Docker Container," *PROSISKO J. Pengemb. Ris.*

dan Obs. Sist. Komput., vol. 7, no. 2, 2020, doi: 10.30656/prosisko.v7i2.2520.

[17] A. R. Ekaputra and A. S. Affandi, "Pemanfaatan layanan cloud computing dan docker container untuk meningkatkan kinerja aplikasi web," *J. Inf. Syst. Appl. Dev.*, vol. 1, no. 2, pp. 138–147, 2023, doi: 10.26905/jisad.v1i2.11084.

[18] S. Applebaum, T. Gaber, and A. Ahmed, "Signature-based and Machine-Learning-based Web Application Firewalls: A Short Survey," *Procedia CIRP*, vol. 189, no. 2019, pp. 359–367, 2021, doi: 10.1016/j.procs.2021.05.105.

[19] I. Gusti, N. Bagus, D. Wiradyaksa, N. Bogi, and A. Karna, "Desain Dan Implementasi Web Application Firewall Dan Rate Limiting Untuk Cyber Defense," *eProceedings Eng.*, vol. 11, no. 6, pp. 1–5, 2024.

[20] G. H. A. Kusuma, "Perancangan Skema Sistem Keamanan Jaringan Web Server menggunakan Web Application Firewall dan Fortigate untuk Mencegah Kebocoran Data di Masa Pandemi Covid-19," *J. Informatics Adv.*, vol. 2, no. 2, pp. 1–4, 2021.

[21] R. Laipaka, "Menerapkan Teknik Firewall Aplikasi Web (WAF) Pada Aplikasi SINTEL Untuk Mengatasi Serangan Siber," *Pros. Semin. Nas. Inov. dan Adopsi Teknol.*, vol. 4, no. 1, pp. 1–10, 2024, doi: 10.35969/inotek.v4i1.407.

[22] S. Karanam, "Ransomware detection using windows API calls and machine learning," Ph.D. dissertation, Virginia Tech, 2023.

[23] M. Encep, A. Hidayatullah, H. Hidayat, M. Z. I. Fauzi, and N. A. Syafitri, "Implementasi Sistem Operasi Server Linux Ubuntu untuk Server NAS menggunakan TRUENAS," *Karimah Tauhid*, vol. 3, no. 10, pp. 11338-11346, 2024.

[24] D. Kartika, R. Riska, and Y. Mardiana, "Dns Server And Web Server Simulation With Debian Operating System On Local Area Network," *J. Media Comput. Sci.*, vol. 2, no. 1, pp. 83–92, 2023, doi: 10.37676/jmcs.v2i1.3439.

[25] Reza. Aditama; Edi. Negara, "Pemindai Kerentanan Terhadap Website Jago Masak Dengan Metode Pengujian Penetrasi OWASP ZAP," *J. Mantik*, vol. 6, no. 3, pp. 3406–3412, 2022.

[26] A. Subari, S. Manan, E. Ariyanto, and A. Fauzi, "Pemanfaatan Metode Wavs (Web Application Security Scanners) Menggunakan Burp Suite Tools Dalam Audit Teknis Keamanan Sistem Informasi Surat Tugas Sekolah Vokasi Undip," *Gema Teknol.*, vol. 21, no. 4, pp. 125–130, 2021, [Online]. Available: http://st2.vokasi.undip.ac.id

[27] R. T. Fielding, M. Nottingham, and J. Reschke, "RFC 9110: HTTP Semantics," no. c, pp. 1–194, 2022.

[28] S. Dhote, S. Singh, D. D. Raigar, and A. Magdum, "A Comprehensive Survey of ML-Based WAFS with Signature and Anomaly Detection," *Strad Research*, vol. 11, no. 4, pp. 54-59, April 2024.