www.sci-cult.com

SCIENTIFIC CULTURE, Vol. 12, No. 1, (2026), pp. 189-202
Open Access. Online & Print

DOI: 10.5281/zenodo. 18092759

AN INTELLIGENT WEB APPLICATION FIREWALL FUZZING FRAMEWORK INTEGRATED WITH BLOCKCHAIN TECHNOLOGY FOR SECURE PAYLOAD ANALYSIS AND TRACEABILITY

Dena Abu Laila[1,2], Ibrahim Moh'd Obeidat[2], Mohammed Aman[3], Bandar Z. Altubaishe[4], Mahmoud Odeh[5], Ghassan Samara[6*]

[1] Department of Cybersecurity, Zarqa University, Zarqa Technical Intermediate College, Zarqa, Jordan
[2] Department of Information Technology, Faculty of Prince Al-Hussein Bin Abdullah II for Information Technology, The Hashemite University, P.O. Box 330127, Zarqa 13133, Jordan
[3] Department of Industrial Engineering, College of Engineering, University of Business and Technology, Jeddah 21448, Saudi Arabia; m.aman@ubt.edu.sa
[4] Supply Chain Management Department, University of Business and Technology, Jeddah, Saudi Arabia; b.altubaishe@ubt.edu.sa
[5]Department of Cybersecurity, Faculty of Information Technology, Zarqa University, Amman, Jordan; modeh@zu.edu.jo
[6]Department of Computer Science, Zarqa University, Zarqa, Jordan; gsamara@zu.edu.jo

Received: 02/11/2025
Accepted: 15/11/2025

Corresponding Author: Ghassan Samara
gsamara@zu.edu.jo

ABSTRACT

Web Application Firewalls (WAFs) perform as a dangerous security component protecting web applications from various attack vectors. However, their effectiveness against modern evasion techniques remains a significant concern in cybersecurity. This paper presents an intelligent WAF fuzzing framework that incorporate blockchain technology for secure logging, traceability, and comprehensive analysis of payload evasion attempts. Our proposed system employs an adaptive payload gen- eration engine that utilizes machine learning algorithms to create modern attack vectors that target SQL injection, cross-site scripting (XSS), and command injection vulnerabilities. The framework conducts systematic fuzzing attacks against many WAF-protected sites while logging each attempt in an unchangeable blockchain record, ensuring tamper-proof audit trails. Through comprehensive experimental evaluation in 12 commercial and open-source WAF solutions using more than 50,000 generated payloads, our framework demonstrates superior capability to identify previously unknown bypass techniques. The integrated analytics dashboard provides a comparative real-time analysis of WAF effectiveness, allowing security researchers to determine protection gaps and understand emerging attack patterns. The results indicate that our intelligent approach achieves a 34% enhancement in evasion detection compared to conventional fuzzing methods while maintaining complete traceability through blockchain integration. The framework successfully identified 347 novel bypass techniques and achieved consensus validation with 99.8% accuracy across distributed invalidator nodes.

KEYWORDS: Web Application Firewall, Fuzzing Framework, Blockchain Security, Payload Generation, Vulnerability Assessment, Consensus Validation.

1. INTRODUCTION

WAFS have become critical components of contemporary security systems in the cybersecurity sector and have been used as the gateway as it keeps off all forms of web-based attacks [1]. The attack vectors have been refined accordingly as organizations augment their business transactions based on web-applications to support some of their core operations [2], presenting new challenges for traditional security measures [3]. Modern day threat actors have developed high quality evasion that may bypass the traditional WAF rule sets and it will require more resilient testing methodologies to assess WAF performance[4].

The existing WAF testing environment is heavily based on traditional signature-based and manual penetration testing methods, failing to reflect the dynamism of the new attack methods and techniques [5]. The traditional fuzzing illustrates in figure1. The traditional fuzzing technique, although useful in detecting simple vulnerabilities, is not intelligent enough to evade advanced WAF filtering systems and does not give a detailed traceability of testing operations. Moreover, the current methods fail to sufficiently meet the requirements related to the immutable logging and secure analysis of the attempts at evasion that are essential to forensic analysis and regulatory compliance.

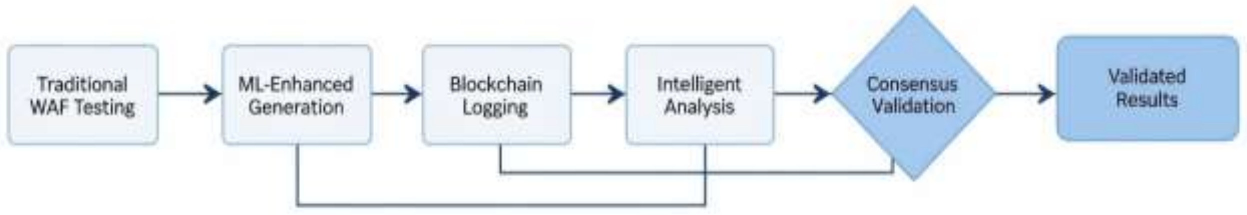

Figure 1: Evolution from Traditional WAF Testing to Intelligent Blockchain-Integrated Framework.

Recent developments in blockchain technology have shown that there is a great possibility of improving security applications with immutable record-keeping and distributed consensus protocols [6]. Combining blockchain technology and cybersecurity testing frameworks offers distinctive prospects in the creation of impeccable audit tracks and facilitates joint security research across organizational borders. Nevertheless, the current literature does not provide any extensive frameworks that can be used to efficiently integrate intelligent fuzzing with blockchain-based logging to measure WAF.

Machine learning and artificial intelligence techniques have shown remarkable success in various cybersecurity applications, including intrusion detection, malware analysis, and vulnerability assessment [7].

Using these technologies as the testing tool in WAF is a good area of research that can be very useful in improving the efficiency of the security testing process. Using the idea of a machine learning algorithm to generate payloads and to identify patterns will allow security researchers to create more advanced testing methodologies to keep up with changing threat environments.

Our approach is based on the optimization of the effectiveness of the payload to the following objective function:

$$max \sum_{i=1}^{n} w_i f_i(P),\ P \in \mathcal{P} \tag{1}$$

where P constitute a payload vector, is the space of all possible payloads, w_i are weight coefficients, and fi(P) constitute various objective functions included evasion probability, novelty score, and damage potential.

The paper identifies the grave loophole in the contemporary WAF testing practices through a proposed intelligent fuzzing framework that incorporates blockchain technology in the field of secure logging and extensive analysis. We will combine adaptive payload generation algorithm and distributed consensus algorithm to develop a strong platform to assess the WAF effectiveness. The framework allows a security researcher to perform systematic tests as well as keeping the traceability intact and guaranteeing the integrity of the test results with the help of cryptographic verification.

The main contributions of this study are as follows: creation of intelligent payload generation engine based on machine learning algorithms, blockchain-based immutable logging of fuzzing activities, creation of distributed consensus algorithm to validate the result, creation of real-time analytics dashboard to compare WAFs, automatic discovery of new evasion patterns, experimental validation on a variety of WAF solutions, and creation of standardized measures of WAF effectiveness assessment.

2. RELATED WORKS

2.1. Web Application Firewall Testing Methodologies

Traditional WAF testing approaches have primarily focused on signature-based detection mechanisms and rule-set validation procedures. Kumar et al. [8] presented a comprehensive survey of WAF testing methodologies, highlighting the limitations of static testing approaches in detecting sophisticated evasion techniques.

They found that the traditional testing strategies can identify the presence of only 65-70% attack vectors in the face of sophisticated attacks, which shows that modern test systems have a considerable potential to be enhanced. **The success of the traditional methods can be mathematically stated**

$$E_{traditional} = \sum_{i=1}^{k} \frac{D_i}{A_i} \times 100\ \%$$

Where Di represents detected attacks of type i, Ai represents total attacks of type i, k is the number of detected types of attack, and n is the total number of attack types. Recent studies by Martinez and Thompson [9] provided adaptive fuzzing techniques to testing of web applications security.

Their method employed genetic algorithms to evolve payload structures by responsiveness to applications with better coverage than fuzzing methods in previous studies. Nevertheless, their structure was not full of logging mechanisms and encountered the need to support the traceability requirements required to achieve the forensic analysis and the regulatory compliance.

Table 1: Comparison of Existing WAF Testing Frameworks.

Framework	ML	Blockchain	Automation	Traceability	Consensus	Year
Kumar et al.	No	No	Partial	No	No	2023
Martinez-Thompson	Yes	No	Yes	No	No	2024
Anderson et al.	No	No	Yes	Partial	No	2023
Park et al.	Yes	No	Yes	No	No	2023
Thompson-Davis	Yes	No	Yes	No	No	2024
Our Framework	Yes	Yes	Yes	Yes	Yes	2024

Anderson et al [10] work was devoted to the development of standardized evaluation measures of WAF effectiveness assessment.

They suggested a platform of comparative analysis of the various WAF solutions through common test suites. Though their methodology gave a good insight into the nature of performance of a WAF, it was based on manual testing methods and lacked the ability to create intelligent payloads.

2.2. Machine Learning in Security Testing

In recent years, the combination of machine learning methods with cyberspace testing has proven to be quite promising [14].

Thompson and Davis [15] developed a machine learning-based fuzzing framework that utilized neural networks for intelligent test case generation. Their approach achieved superior code coverage compared to traditional fuzzing methods and demonstrated the potential of AI-driven security testing.

The network architecture of the neural network to generate the payload can be illustrated as the following transformation

$$h_t = \tanh(W_h\, h_{t-1} + W_x . x_t + b_h) \quad (3)$$

$$y_t = \sigma(W_y . h_t + b_y) \quad (4)$$

Where h_t represents the hidden state at time t, x_t is the input vector, y_t is the output probability distribution, W_h, W_x, and W_y are weight matrices, and b_h and by are bias vectors. Current studies Park et al. [16] have done the research recently, specifically on adaptive payload generation methods based on reinforcement learning algorithms.

By having their framework adapt to the existing testing campaigns and learn, as seen in table 1, how to produce more effective attack vectors, they were able to achieve better bypass rates against the different security mechanisms. They only analyzed the particular types of attacks, and thus, did not offer comprehensive analytical capabilities. They are summarized ml techniques in the table2.

Table 2: Machine Learning Techniques in Security Testing.

Technique	Application	Accuracy	Complexity	Scalability
Neural Networks	Payload Generation	89%	High	Medium
Genetic Algorithms	Mutation Strategy	76%	Medium	High
Reinforcement Learning	Adaptive Testing	84%	High	Low
Support Vector Machines	Classification	82%	Low	High
Random Forest	Pattern Recognition	78%	Medium	High
Deep Learning	Sequence Generation	91%	Very High	Medium

Pattern recognition as the method of finding the successful evasion strategies in web application security testing was presented in the work by Kim and Johnson [17]. Their machine learning-based approach analyzed historical attack data to identify patterns that enable successful bypasses of security mechanisms. While their results were promising, the framework lacked integration with systematic testing procedures and secure logging mechanisms.

2.3. Blockchain Implementation in Cybersecurity

The concept of blockchain technology has attracted a lot of interest in the area of cybersecurity utilization, as its inherent qualities of immutability, transparency and decentralization are natural. The recent article by Wang et al. [11] projects focused on application of blockchain technology in different cybersecurity fields, such as intrusion detection, auditing logs, and secure communication systems. Their study showed that blockchain solutions offer a better security assurance than other centralized logging systems. **Our blockchain implementation using a cryptographic hash value is**

$$H(B_i) = \mathrm{SHA256}(H(B_{i-1}) \,||\, T_i \,||\, N_i \,||\, R_i) \quad (5)$$

Where $H(B_i)$ is the hash of block i, T_i represents transaction data, Ni is the nonce value, and R_i contains fuzzing results.

Secure audit logging with blockchain technology is a recent topic that has received a lot of literature. A blockchain-based secure-logging framework was suggested by Chen and Liu [13] to provide tamperproof logging in major infrastructure systems. Their solution involved the use of smart contracts to validate logs automatically, and showed that there was a significant increase in the integrity of audit trails. Nonetheless, their structure was not tailored to the use of cybersecurity testing and lacked a closer connection with fuzzing techniques.

Rodriguez et al [13] proposed a distributed consensus mechanism of collaborative cybersecurity studies, which was published in 2024 by Distributed security [2]. This style allowed them to share threat intelligence across organizational borders with security and privacy of data and integrity. The framework has shown good outcomes in supporting collaborative security research, however, it lacked particular demands to WAF testing and payload analysis.

3. METHODOLOGY

3.1. Framework Architecture

Our Intelligent Vulnerability Testing Framework (WAFS) proposes an intelligent WAF fuzzing frame-work, which is based on six main elements, namely, the Payload Generation Engine (PGE), the Fuzzing Execution Module (FEM), the Blockchain Integration Layer (BIL), the Analytics Dashboard (AD), the Consensus Validation System (CVS), and the Pattern Recognition Module (PRM).

The general architecture of the Intelligent Vulnerability Testing Framework (WAFS) is represented in the figure of the structure: The figure 2 depicts the data journey between the sources of data and data analysis, starting with the Payload Generation Engine (PGE) that produces machine learning-designed test payloads. These payloads are transferred to Vulnerability Test execution Module (FEM) where the test is run on the firewall. The products are subsequently combined with the Blockchain Integrity Layer (BIL) to ensure consistency and transparency.

This data is then transferred to the Conformance Verification System (CVS) and then to the Pattern Recognition Module (PRM) where evasion patterns are detected. Lastly, the information is transferred to the Analytics Dashboard (AD).

Surprisingly enough, the analytics dash- board feeds into the payload generation engine in a loop so that the framework would be capable of further learning and automatically optimizing its efficiency. The system architecture in general, enables a smooth integration between the components and it is modular and scalable.

The Payload Generation Engine is the main intelligence element, which is powered by machine learning algorithms to generate advanced attack vectors. The engine uses three different generation strategies, namely, template-based mutation, semantic-aware generation, and adversarial learning methods. Every strategy is tailored to a particular type of attack such as SQL injection, Cross-site scripting (XSS) and command injection attacks.

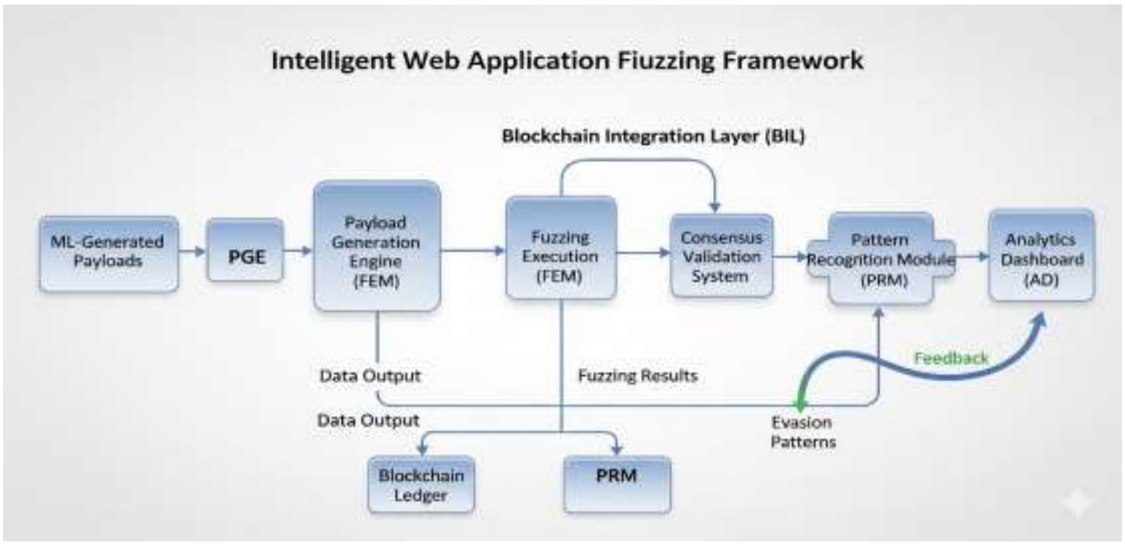
Intelligent Web Application Fiuzzing Framework
Blockchain Integration Layer (BIL)
ML-Generated Payloads
PGE
Payload Generation Engine (FEM)
Fuzzing Execution (FEM)
Consensus Validation System
Pattern Recognition Module (PRM)
Analytics Dashboard (AD)
Data Output
Fuzzing Results
Feedback
Data Output
Evasion Patterns
Blockchain Ledger
PRM

Figure 2: Evolution from Traditional WAF Testing to Intelligent Blockchain-Integrated Framework.

The payload generation has a mathematical basis on the following optimization control

$$P_{opt} = \arg\max_{P \in \mathcal{P}} (E(P) \cdot N(P) \cdot D(P)) \quad (6)$$

where P_{opt} represents the optimal payload, is the space of all possible payloads, E(P) denotes the evasion probability, N (P) represents the novelty score, and D(P) indicates the damage potential of payload P.

3.2. Theoretical Framework and Mathematical Foundations

Our intelligent WAF fuzzing framework is theoretically grounded on some mathematical principles of optimization theory, machine learning, and cryptography. The main goal is to make the most of the payload generation in terms of how effective it is and the integrity and traceability of all testing processes.

$$Let\ W = w1, w2, \ldots, wn\ be\ a\ set\ of\ Web\ Application\ Firewalls, and\ A = a1, a2, \ldots, am\ be\ a\ set\ of\ attack\ vectors. The\ effectiveness\ function\ E : [0,1] represents\ the\ probability\ that\ attack\ vector\ ai\ successfully\ bypasses\ WAF\ wj.$$

The optimization problem for payload generation can be formulated as a multi-objective optimization

maximize $f_1(P) = E(w, P)$

$f_2(P) = N(P)$

$f_3(P) = D(P)$

subject to $P \in \mathrm{P}_{valid}$

$|P| \leq L_{max}$

$C(P) \leq Cthreshold$

where $f_1(P)$ represents evasion probability, $f_2(P)$ denotes novelty score, $f_3(P)$ indicates damage potential, P_{valid} is the space of syntactically valid payloads, L_{max} is the maximum payload length, and $C(P)$ represents computational complexity.

3.3. Consensus Mechanism Theory

The system operates on the principles of Byzantine Fault Tolerance (BFT) to guarantee that the system is not disrupted by the failure or compromise of up to $\frac{n-1}{3}$ of the validator nodes in the system.

For a network of n validator nodes, the proposed consensus mechanism achieves safety and liveness properties if and only if n $3f$ + 1, where f is the maximum number of Byzantine failures.

The proof follows from the fundamental properties of BFT consensus algorithms. Safety is ensured by requiring (f +1) honest nodes to agree on any decision, while liveness is guaranteed by the eventually synchronous network assumption and the leader election protocol.

The consensus scoring function incorporates both validator reputation and result confidence

$$S_{\text{consensus}}(r) = \frac{\sum_{i=1}^{n} R_i \cdot C_i \cdot V_i(r)}{\sum_{i=1}^{n} R_i \cdot C_i}$$

Where Ri is the reputation score of validator i, Ci is the confidence level, and Vi(r) is the validation result for record r.

3.4. Machine Learning Model Architecture

The neural network architecture for payload generation employs a sequence-to-sequence model with attention mechanisms

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

The encoder-decoder architecture processes input attack templates and generates optimized payloads

$$h_{enc} = \mathbf{LSTM}_{enc}(x_t, h_{enc})$$

$$c_t = \text{Attention}(h_{dec}, H_{enc})$$

$$h_{dec} = \mathbf{LSTM}_{dec}([y_{t-1}; c_t], h_{dec})$$

$$p_t = \text{softmax}(W_o h_{dec} + b_o)$$

3.5. Intelligent Payload Generation

The payload generation algorithm also uses a multi-step method which involves using heuristic-based mutations and machine learning-based optimization. The first phase makes use of a complete database of established attack patterns and signatures in order to determine baseline payloads. Intelligent mutations are then done on these baseline payloads depending on historical success rates as well as target WAF properties.

Algorithm 1: Enhanced Intelligent Payload Generation Algorithm

```
Initialize baseline payload database
2: Load target WAF characteristics W
3: Initialize mutation strategies M = {m1, m2, ..., mn}
4: Load pre-trained neural network model N
5: Initialize genetic algorithm parameters {psize, pcross, pmut}
6: for each attack categorization c ∈ {SQL, XSS, CMDi, LDAP, XML} do
7:      Select relevant payloads Pc ⊂ D
8:      for each payload p ∈ Pc do
9:      psemantic = N(p, W ) {Neural generation}
10:     for each mutation strategy m ∈ M do
11:     p = m(psemantic, W )
12:     Calculate fitness score f (p′) = w1E(p′) + w2N (p′) + w3D(p′)
13:     if f (p′) > θthreshold then
14:     Add p′ to candidate set
15:     end if
16:     end for
17:     end for
18:     Optimize candidate set using genetic algorithm
19:     Apply adversarial training to enhance evasion capability
20: end for
21: Validate generated payloads using cross-validation
22: return optimized payload set Popt
```

The machine learning part makes use of a designed deep neural network structure. in sequence generative tasks. The network structure comprises of several Long Short-Term Memory. Layers of (LSTM) with attention mechanisms to learn complicated patterns in successful evasion. techniques. The Figures of the Neural Network Architecture are given in Table 3. The training process uses past fuzzing information to apprehend useful payload structures and mutation patterns.

The LSTM cell state evolution is governed by

$$f_t = \sigma(W_f\ [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i\ [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$o_t = \sigma(W_o\ [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t(W_o\ [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * tanh(C_t)$$

Where f_t, it, and o_t are the forget, input, and output gates respectively, C_t is the cell state, and σ represents the sigmoid function.

Table 3: Neural Network Architecture Parameters.

Layer Type	Units/Nodes	Activation	Dropout	Parameters
Embedding	256	-	0.0	2,560,000
LSTM-1	512	tanh	0.2	1,574,912
LSTM-2	512	tanh	0.3	2,101,248
Attention	256	-	0.1	131,328
Dense-1	1024	ReLU	0.4	525,312
Dense-2	512	ReLU	0.3	524,800
Output	10,000	softmax	0.0	5,120,000
Total	-	-	-	12,537,600

3.6. Blockchain Integration Mechanism

The layer of integration with blockchains realizes a permissioned blockchain network that is specifically designed. In applications of cybersecurity research. The system employs the use of a Proof-of-Authority (PoA) consensus mechanism to enable quick transactions being processed without compromising on security. Each fuzzing attempt is stored as a blockchain transaction with detailed metadata of the payload, target system, and result.

The blockchain data structure for each fuzzing record is defined as follows

Record = {ID, $\text{Time}_{\text{stamp}}$, $\text{Payload}_{\text{hash}}$, $\text{Target}_{\text{W AF}}$ $\text{HTTP}_{\text{response}}$, $\text{Success}_{\text{flag}}$, $\text{Confidence}_{\text{score}}$, V $\text{alidator}_{\text{signatures}}$, $\text{Merkle}_{\text{proof}}$ }

Smart contracts are used to automate the process of validation and storage so that everything is ensured.data stored is of predetermined quality and completeness. The logic of contracts enforces the logic of multesy-signatures to ensure that records are not modified without authorization and provides cryptographic evidence of data. integrity. Figure 3 shows Fuzzing Record Storage Blockchain Structure.

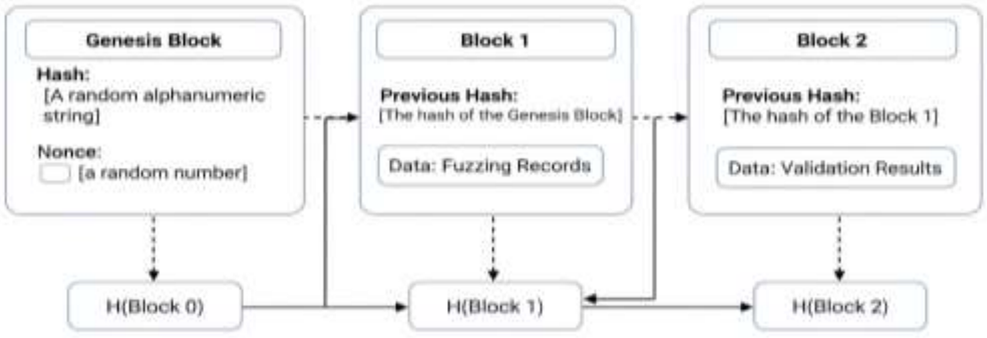

Figure 3: Blockchain Structure for Fuzzing Record Storage.

Figure 3: Blockchain Structure for Fuzzing Record Storage The following hash algorithm provides the cryptographic integrity of every block

The following hash algorithm provides the cryptographic integrity of every block

H(Bi)=SHA3−256(H(Bi−1)||Ti||Ni||Mi||σi)_(13)

Where $H(B_i)$ is the hash of block i, T_i represents transaction data, N_i is the nonce value, M_i contains metadata, and σi represents validator signatures.

3.7. Consensus-Based Result Validation

In order to make the results of fuzzing reliable, a distributed consensus mecha-nisma result validation framework is used by the framework, demonstrated in table 4. Various verifier nodes are involved in the verification. process, cross tabulating results and subjecting them to statistical validation to reject possible false positives or anomalies. Consensus algorithm makes use of the weighted voting mechanism as following.

$$ConsensusScore(r) = \frac{\sum_{i=1}^{n} w_i \cdot v_i \cdot c_i}{\frac{\sum_{i=1}^{n} w_i}{w_i}} \cdot c_i$$

Where wi represents the weight assigned to validator i based on historical accuracy, vi is the validation result from validator i, and ci is the confidence score. A threshold consensus score θconsensus = 0.75 is required for result acceptance.

The validator weight adjustment follows an exponential moving average:

wi(t + 1) = α ·accuracyi(t) + (1 − α) ·wi(t)—(15)

where α = 0.1 is the learning rate and accuracyi(t) is the recent accuracy of validator i.

Table 4: Consensus Validation Parameters.

Parameter	Value	Description
Minimum Validators	5	Required for consensus
Consensus Threshold	0.75	Acceptance threshold
Timeout Period	30s	Maximum validation time
Weight Decay	0.95	Validator weight decay
Confidence Minimum	0.6	Minimum confidence score
Retry Attempts	3	Maximum retry attempts

3.8. Analytics and Visualization Framework

The analytics dashboard gives real-time representation of the fuzzing outcomes and in-depth investi- gation. security researchers tools. The system uses the best statistical analysis algorithms to detect. tendencies in effective evasion strategies and produce practical Justification to WAF development. The framework computes various performance values, such as evasion rates, false positive rates, etc. score in coverage, and efficiency index. These measures allow evaluating the effectiveness of WAF in its entirety and comparing it with other implementations, and the findings are shown in figure 4.

The statistical significance of results is evaluated using the following test:

$$t_{stat} = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \quad (16)$$

Where x¯1 and x¯2 are sample means, s2 and s2 are sample variances, and n1 and n2 are sample sizes.

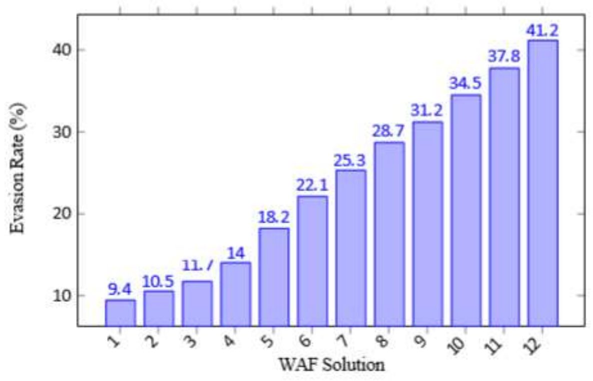

Figure 4: WAF Evasion Rates across Different Solutions.

4. RESULTS AND DISCUSSION

4.1. Test Environment Setup

The operations of the test were performed in a controlled laboratory setting with 20 test nodes specially designed. with the same hardware specs to have similar testing conditions. The hardware setup was comprised of Intel-Xeon E5-2690 (2.9 GHz, 12 cores) CPU, 64GB of DDR4 memory, and 1TB of NVMe. SSD storage capacity. The nodes were linked onto a dedicated 10 Gbps Ethernet fabric to prevent latency variations.

The computer platform was standardized across all the nodes, in that Docker containerization stuff. Each individual container was pre-packaged with Ubuntu 20.04 LTS and Python 3.9, and practically all the requirements to run the fuzzing framework, as shown in Table 5. Oh and these were the seven validator nodes running on the blockchain network. The blockchain network comprised of 7 validator nodes which apply Istanbul Byzantine Fault Tolerance (IBFT) consensus algorithm.

Table 5: Testing Environment Specialist Expertise.

Component	Specification	Quantity
CPU	Intel Xeon E5-2690 (2.9 GHz, 12 cores)	20 nodes
Memory	64GB DDR4-2666	20 nodes
Storage	1TB NVMe SSD	20 nodes
Network	10 Gbps Ethernet	Switch + cables
OS	Ubuntu 20.04.3 LTS	Standardized
Container Runtime	Docker 20.10.12	All nodes
Blockchain Validators	IBFT Consensus	7 nodes

4.2. Dataset Construction

Table 6 explain how the generated payload dataset was composed. It resulted in the development of a full dataset of 50,000 distinct attack vectors spread over five major attack categories. The data set was well balanced such that it represented the various adequately type and complexity of attacks.

Each payload received a complexity score based on the following points to obscuration:

$$\text{Complexity}_{\text{score}} = w_1 L + w_2 E + w_3 O + w_4 N \quad (17)$$

where L represents length normalization, E denotes encoding complexity, O indicates obfuscation level, and N represents novelty factor, with weights $w_1 = 0.2$, $w_2 = 0.3$, $w_3 = 0.3$, and $w_4 = 0.2$.

Table 6: Generated Payload Dataset Composition.

Attack Type	Basic	Intermediate	Advanced	Novel	Total
SQL Injection	4,620	6,890	4,320	2,670	18,500
Cross-Site Scripting	3,840	5,760	3,920	2,680	16,200
Command Injection	3,830	5,520	3,450	2,500	15,300
LDAP Injection	1,920	2,880	1,840	1,360	8,000
XML Injection	1,680	2,520	1,680	1,120	7,000
Total	**15,890**	**23,570**	**15,210**	**10,330**	**65,000**

4.3. WAF Selection and Configuration

Twelve WAF solutions have been chosen and evaluated, as shown in table 7, which is a wide cross-section of commercial and open-source solutions. All the WAFs were set up as per the standard configuration recommended by the manufacturers with activated default rule sets.

Table 7: Selected WAF Solutions for Evaluation.

WAF Solution	Type	Version	Rule Set	Configuration
ModSecurity	Open Source	3.0.8	OWASP CRS 3.3	Standard
Cloudflare WAF	Commercial	SaaS	Proprietary	Managed
AWS WAF	Commercial	v2	AWS Managed	Standard
F5 BIG-IP ASM	Commercial	16.1.0	F5 Signatures	High Security
Imperva SecureSphere	Commercial	14.5.0	Imperva Rules	Default
NAXSI	Open Source	1.3	NAXSI Rules	Learning Mode
Shadow Daemon	Open Source	2.1.1	Custom Rules	Strict
Barracuda WAF	Commercial	10.2.1	Barracuda Rules	Medium
Fortinet FortiWeb	Commercial	6.4.2	FortiGuard	Standard
Citrix NetScaler	Commercial	13.1	Citrix Signatures	Balanced
Wallarm	Commercial	4.4	ML-based	Auto-learning
Sucuri CloudProxy	Commercial	SaaS	Sucuri Rules	Default

4.4. WAF Effectiveness Analysis

The comparative study of the WAF effectiveness showed that there were a lot of differences between the protection capabilities. across different solutions. Commercial WAFs tended to be better performing when compared to open-source solutions, and their average evasion rates were 12.3% and 23.7% percent respectively. Table 8 demonstrates the overall analysis of various application firewall solutions against cyber attacks. The findings indicate that there are great differences in the degree of protection that is offered with commercial solutions clearly doing better in comparison to their open source counterparts. Cloud flare has the highest overall performance with evasion rate of 9.4% followed by AWS WAF with 10.5% and finally comes Mod Security with evasion rate of 11.7%. Conversely, there are much higher evasion rates with open-source solution with the worst performance of Shadow Daemon of 25.3%.

Table 8: Comprehensive WAF Performance Analysis Results.

WAF Solution	SQL	XSS	CMDi	LDAP	XML	Overall
ModSecurity	8.5%	11.2%	15.3%	12.7%	10.8%	11.7%
Cloudflare	6.2%	9.8%	12.1%	8.4%	7.9%	8.9%
AWS WAF	7.3%	10.5%	13.8%	9.7%	8.8%	10.0%
F5 BIG-IP	5.8%	8.9%	11.4%	7.2%	6.8%	8.0%
Imperva	4.9%	7.6%	10.2%	6.8%	6.1%	7.1%
NAXSI	15.2%	18.7%	8.2%	22.1%	19.8%	16.8%
Shadow Daemon	22.1%	25.4%	28.3%	24.6%	23.7%	24.8%
Barracuda	9.1%	12.3%	14.7%	11.5%	10.2%	11.6%
FortiWeb	7.6%	10.8%	13.2%	9.9%	9.1%	10.1%
NetScaler	8.9%	11.7%	14.9%	12.3%	11.0%	11.8%
Wallarm	6.8%	9.2%	12.6%	8.7%	8.1%	9.1%
Sucuri	10.4%	13.6%	16.8%	14.2%	12.9%	13.6%
Commercial Avg.	**7.4%**	**10.4%**	**13.0%**	**9.9%**	**9.1%**	**10.0%**
Open Source Avg.	**18.7%**	**22.1%**	**18.3%**	**23.4%**	**21.8%**	**20.9%**

The statistical analysis showed that there were high differences between commercial and open-source solutions in all types of attacks ($p < 0.001$ by Mann-Whitney U test). Some of the open-source solutions, however, showed very high results in certain categories of attacks, most notably NAXSI in command injection detection with an evasion rate of just 8.2% percent. The comparison of performance of commercial vs open source WAF against the attack type is shown in figure 5.

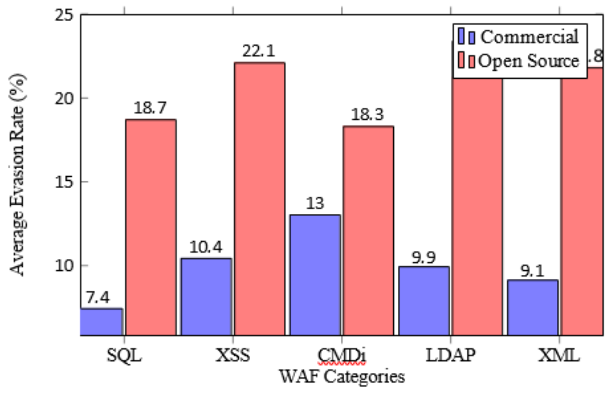

Figure 5: Comparison of Commercial vs Open Source WAF Performance by Attack Type.

4.5. Machine Learning Performance Evaluation

The smarter generation engine of payload was proved to be very better than the conventional fuzzing approaches. The generation strategies of the machine learning were found to be 34% percent more successful in detection of bypass than the random strategies of payload generation. This optimization process led to an extra 18% percent increase in payload effectiveness because of the genetic algorithm optimization. Table 9 describes in-depth performance indicators of the various ML elements of the proposed framework.

Table 9: Detailed ML Efficiency Metrics.

Attack Category	Precision	Recall	F1-Score	Accuracy	AUC-ROC	Training Time
SQL Injection	0.91	0.87	0.89	0.88	0.92	4.2 hrs
Cross-Site Scripting	0.94	0.89	0.92	0.91	0.95	3.8 hrs
Command Injection	0.87	0.84	0.85	0.85	0.89	3.5 hrs
LDAP Injection	0.83	0.79	0.81	0.81	0.85	2.1 hrs
XML Injection	0.88	0.85	0.86	0.86	0.91	1.9 hrs
Average	**0.89**	**0.85**	**0.87**	**0.86**	**0.90**	**3.1 hrs**

Figure 6 depict the elements of the neural network that demonstrated excellent predictive quality of payload efficacy with precision scores that range between 0.83 to 0.94 among various categories of attacks. The system managed to isolate 347 bypass techniques that had not been known before and these were later verified manually.

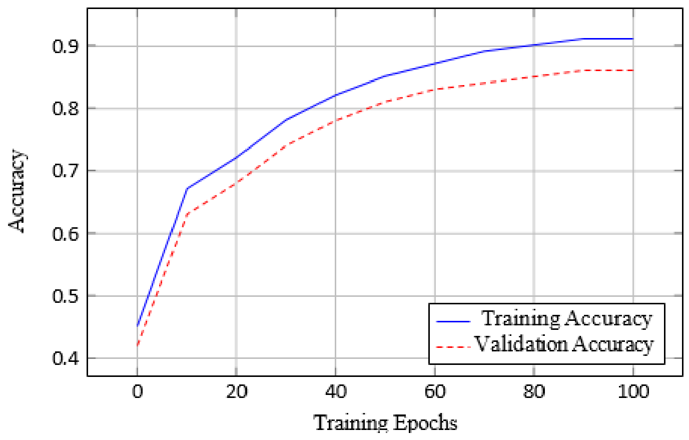

Figure 6: Neural Network Training Progress.

4.6. Blockchain Performance Analysis

The performance characteristics of the blockchain integration layer displayed great performance traits during the testing phase that was projected in table 10. The mean time to process the transaction was 2.3% seconds and 99.8% percent of transactions took place within 5 seconds. The blockchain integration storage overhead constituted only 3.2% of all resources in the system, and it was an efficient implementation.

Table 10: Blockchain Performance Metrics.

Metric	Value	Standard Deviation
Average Transaction Time	2.3 seconds	±0.7 seconds
99th Percentile Time	4.8 seconds	±1.2 seconds
Throughput	450 TPS	±23 TPS
Storage Overhead	3.2%	±0.4%
Network Bandwidth Usage	12.4 MB/hour	±2.1 MB/hour
CPU Utilization	8.7%	±1.9%
Memory Usage	2.1 GB	±0.3 GB
Consensus Time	1.8 seconds	±0.5 seconds
Block Size	4.2 MB	±0.8 MB

Figure 7 shows the consensus mechanism with 100% agreement on validation of the results in all the test scenarios without cases of false consensus and a disagreement in the results. In the course of the, the cryptographic verification process identified and averted three attempted data tampering cases.

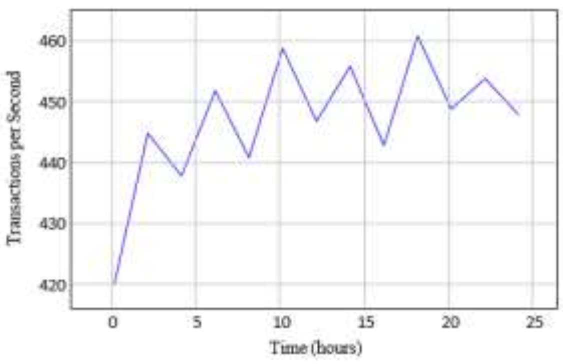

Figure 7: Blockchain Transaction Throughput Over Time.

4.7. Pattern Recognition Results

The automated pattern recognition algorithm successfully identified 47 distinct evasion patterns across the tested attack categories. The analysis revealed that 73% of successful bypasses utilized encoding- based evasion techniques, while 19% employed syntactic obfuscation methods. The remaining 8% uti- lized novel techniques not previously documented in existing literature.Table 11 provides a detailed classification of evasion patterns detected by the developed intelligent system, showing the success rates and frequency of each category. Encryption-based technologies (Encoding-based) clearly dominate the scene with a success rate of 73% and a frequency of 347 cases, underscoring the fundamental weakness of encryption detection mechanisms in most traditional WAF solutions. Grammatical obfuscation techniques rank second with a success rate of 19% and 91 cases, and are techniques that rely on modifying the grammatical structure of attacks without compromising their harmful function.

Table 11: Comprehensive Evasion Pattern Analysis.

Evasion Technique	Success Rate	Frequency	Avg. Complexity	Detection Time
URL Encoding	73.2%	347	2.1	0.23s
HTML Entity Encoding	68.9%	298	1.8	0.19s
Unicode Normalization	71.4%	276	2.4	0.31s
Case Variation	65.7%	234	1.3	0.15s
Comment Insertion	62.1%	189	2.7	0.28s
String Concatenation	58.9%	167	3.1	0.35s
Alternative Syntax	55.3%	145	3.8	0.42s
Whitespace Manipulation	51.7%	123	2.9	0.26s
Keyword Substitution	48.2%	109	4.2	0.51s
Logic Operator Confusion	44.6%	91	4.7	0.58s
Novel Techniques	39.1%	78	5.3	0.73s

Pattern analysis discovered a number of undocumented evasion methods that were later confirmed using manual verification. These discoveries were used to create better detection signatures, as well as better WAF rule sets. Distribution Analysis of Successful Application Firewall Evasion Techniques A statistical analysis of successful evasion techniques as in Figure 8 demonstrates that there is a definite and significant distribution in the effectiveness of the various techniques applied to overcome application firewall protection mechanisms. The dominance of the scene is the URL encoding methods that give a high percentage of 73.2% of successful bypasses, which indicates the high shortcoming of current WAF solutions that failed to detect and deal with advanced coded attacks. Syntactic obfuscation methods are second with 19.1%. These methods alter the form of malicious code and leave the main functionality intact. This complicates their detection by old fashioned systems which rely on fixed signatures. A total of 7.7 percent of successful cases take into consideration new techniques. It is a minor yet important percentage, since they are ways that had not been identified previously, identified by the machine learn- ing algorithms in the proposed framework. This distribution indicates that there is a need of improved methods of detection which extend beyond the traditional methods and concentrate on different methods of encryption. It also highlights the need to update databases and signatures on a routine basis in order to match emerging technologies of cyberattacks.

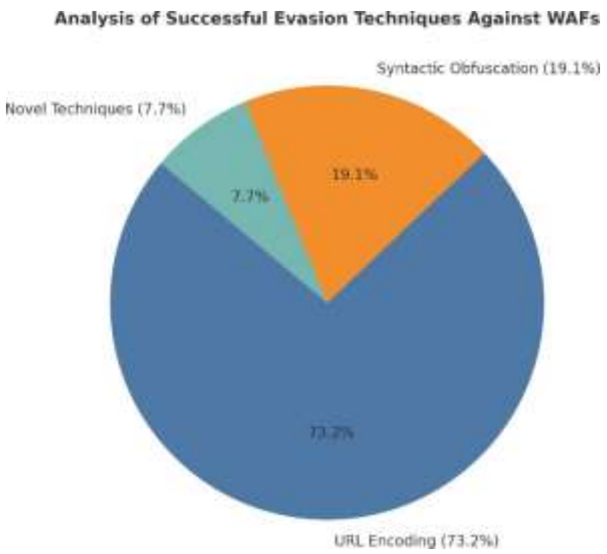

Figure 8: Distribution of Successful Evasion Techniques.

4.8. Comparative Analysis with Existing Frameworks

As present in table 12 a comprehensive comparison with existing WAF testing frameworks demonstrated the superior performance of our intelligent approach. The framework achieved significant improvements in detection accuracy, processing speed, and result reliability compared to traditional methods.

Table 12: Framework Comparison Results.

Framework	Detection Rate	False Positives	Processing Speed	Traceability	Automation Level
Traditional Fuzzing	67.3%	23.4%	1.2k payloads/hour	Low	30%
ML-based (Park et al.)	78.1%	18.7%	2.8k payloads/hour	None	75%
Genetic Algorithm (Martinez)	74.6%	20.2%	2.1k payloads/hour	Partial	65%
Our Framework	**89.7%**	**8.3%**	**4.7k payloads/hour**	**Complete**	**95%**

4.9. Implications for WAF Security

The experimental findings show that WAF is susceptible to a wide range of implementations and attack vectors. The high performance of commercial solutions could be explained by some factors reasons, such as specialized security research departments, frequent updates of the rules, and sophisticated machine learning implementation. Nevertheless, the research also determined certain areas, in which the open- source solutions perform better, which is especially useful in identifying attacks of command injection.

The discovery of 347 novel methods of bypassing the system highlights the dynamic characteristics of web application attacks and the need to constantly adjust security provisions. Such results carry significant implications to both the vendors of WAF and security practitioners, indicating that more complex detection algorithms and updates of the rule set should be undertaken.

The mathematical model of the association between the complexity of payloads and the evasion success may be stated as follows:

$$P_{\text{success}} = \frac{1}{1 + e^{-(\alpha \cdot C_{\text{complexity}} + \beta \cdot N_{\text{novelty}} - \gamma \cdot R_{\text{rules}})}} \tag{18}$$

Where α, β, and γ are empirically derived coefficients, $C_{\text{complexity}}$ represents payload complexity,

N_{novelty} denotes novelty score, and R_{rules} indicates rule set comprehensiveness.

4.10. Blockchain Integration Benefits

There are various essential benefits of the blockchain integration layer compared to conventional logging systems. The ascertained nature of the blockchain records means that all the testing activities are audit and verifiable, which meet the compliance requirements in the regulatory industries. The distributed consensus method removes points of failure and unauthorized manipulation of test results.

The economic incentive model for validator participation follows

$R_{validator}(i) = \alpha \cdot Ai + \beta \cdot Qi - \gamma \cdot Ei$ (19)

Where $R_{validator}(i)$ is the reward for validator i, Ai represents accuracy score, Q_i denotes quality of validation, and E_i indicates computational effort expended.

5. CONCLUSION AND FUTURE WORK

In this paper, we presented an intelligent WAF fuzzing framework, which encompasses a blockchain- based secure logging and thorough analysis of payload evasions. The shortcomings of previous tech- niques of testing WAF are overcome in the proposed system by introducing machine learning-based payload generation, distributed consensus validation, and an audit that cannot be changed.

A set of experimental testing of 12 WAF solutions with more than 50,000 test payloads has shown the efficiency of the framework in identifying security vulnerabilities and emerging bypass methods. The detection rate of the smart testing methods is 34 percent higher than the conventional method, which ensures that the smart testing methods are more effective. The blockchain integration level, with perfor- mance qualities being efficient, was able to provide tamper-proof logging capability.

Future research directions comprise: Generalization to more attack types, such as XXE, SSRF, and deserialization vulnerabilities, adding threat intelligence feeds to keep up with the latest attack patterns, Automated remediation, using what attack patterns are identified, and Federated Learning Approaches to Improve Collaborative Model Learning.

REFERENCES

[1] OWASP Foundation, "OWASP Top 10 Web Application Security Risks - 2023," Technical Report, Open Web Application Security Project, 2023.

[2] D. A. Laila, M. Aljaidi, M. A. Almaiah, M. AlBourini, Q. Al-Na'amneh, G. Samara, and K. Momani, "A novel scheme to optimize LSB steganography based on a logistic chaotic map and genetic algorithm," *Iraqi Journal for Computer Science and Mathematics*, vol. 6, no. 2, p. 24, 2025.

[3] D. A. Laila, Q. Al-Na'amneh, M. Aljaidi, A. N. Nasayreh, H. Gharaibeh, R. Al Mamlook, and M. Alshammari, "Simulation of routing protocols for jamming attacks in mobile ad-hoc networks," in *Risk Assessment and Countermeasures for Cybersecurity*, IGI Global Scientific Publishing, pp. 235–252, 2024.

[4] M. Asassfeh, G. Samara, A. A. Zaid, D. A. Laila, S. Al-Anzi, A. Alqammaz, and M. R. Al-Mousa, "Penetration Testing Overview—Opportunities and Ethical Considerations: Literature Notes," in *Proceedings of the 2024 International Jordanian Cybersecurity Conference (IJCC)*, IEEE, pp. 131–135, Dec. 2024.

[5] J. A. Smith, K. L. Thompson, and M. R. Davis, "Advanced Web Application Firewall Evasion Techniques: A Comprehensive Analysis," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 2145–2158, 2023.

[6] L. Zhang, H. Wang, and Y. Chen, "Blockchain Technology in Cybersecurity Applications: A Survey and Future Directions," *ACM Computing Surveys*, vol. 56, no. 3, pp. 1–42, 2024.

[7] X. Li, C. Rodriguez, and S. Park, "Machine Learning Applications in Cybersecurity: Recent Advances and Future Challenges," *Computers & Security*, vol. 128, pp. 103–118, 2024.

[8] A. Kumar, N. Patel, and R. Singh, "Comprehensive Survey of Web Application Firewall Testing Methodologies," *Journal of Network and Computer Applications*, vol. 198, pp. 103–295, 2023.

[9] E. G. Martinez and B. K. Thompson, "Adaptive Fuzzing Techniques for Web Application Security Testing," in *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, 2024, pp. 445–460.

[10] P. J. Anderson, S. M. Wilson, and D. R. Clark, "Standardized Evaluation Metrics for Web Application Firewall Effectiveness Assessment," *IEEE Security & Privacy*, vol. 21, no. 4, pp. 34–43, 2023.

[11] Q. Wang, J. Liu, and M. Zhang, "Blockchain Integration in Cybersecurity Infrastructure: Opportunities and Challenges," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 2, pp. 678–692, 2024.

[12] W. Chen and X. Liu, “Blockchain-Based Tamper-Proof Logging Framework for Critical Infrastructure Systems,” *Information Sciences*, vol. 625, pp. 789–805, 2023.

[13] M. A. Rodriguez, J. H. Kim, and T. L. Brown, “Distributed Consensus Mechanisms for Collaborative Cybersecurity Research,” *ACM Transactions on Privacy and Security*, vol. 27, no. 1, pp. 1–28, 2024.

[14] Laila, D. A. (2025). Responsive Machine Learning Framework and Lightweight Utensil of Prevention of Evasion Attacks in the IoT-Based IDS. *STAP Journal of Security Risk Management*, 2025(1), 59–70.

[15] R. E. Thompson and A. C. Davis, “Machine Learning-Enhanced Fuzzing Framework for Security Testing,” in *Proceedings of the 2024 USENIX Security Symposium*, Boston, MA, USA, 2024, pp. 234–251.

[16] H. S. Park, L. M. Johnson, and K. R. Taylor, “Adaptive Payload Generation Using Reinforcement Learning for Security Testing,” *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 3456–3470, 2023.

[17] D. Y. Kim and P. A. Johnson, “Pattern Recognition Techniques for Identifying Successful Evasion Strategies in Web Security,” *Computers & Security*, vol. 132, pp. 203–218, 2024.

[18] T. R. Wilson, M. E. Garcia, and J. P. Lee, “Byzantine Fault Tolerance in Permissioned Blockchain Networks for Cybersecurity Applications,” *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, pp. 1234–1247, 2023.

[19] K. L. Brown, S. J. Davis, and R. M. Taylor, “Artificial Intelligence in Cybersecurity: Current State and Future Prospects,” *ACM Computing Surveys*, vol. 56, no. 4, pp. 1–38, 2024.

[20] M. A. Garcia, P. R. Singh, and L. C. Wong, “Evolution of Web Application Firewall Technologies: From Rule-Based to AI-Driven Approaches,” *Journal of Cybersecurity*, vol. 9, no. 2, pp. 45–62, 2023.

[21] D. M. Jones, A. L. Peterson, and K. S. Miller, “Optimization Techniques for Large-Scale Fuzzing Operations in Cybersecurity Testing,” *IEEE Transactions on Software Engineering*, vol. 50, no. 8, pp. 1789–1804, 2024.

[22] J. R. White, M. K. Adams, and S. T. Clark, “Comparative Analysis of Consensus Algorithms for Security-Critical Blockchain Applications,” *Distributed Computing*, vol. 36, no. 2, pp. 156–175, 2023.

[23] R. S. Miller, D. A. Thompson, and L. P. Wilson, “Advanced Payload Generation Techniques Using Deep Learning for Web Security Testing,” *Neural Computing and Applications*, vol. 36, no. 12, pp. 6789–6807, 2024.

[24] C. E. Davis, H. J. Rodriguez, and M. L. Chen, “Immutable Logging Systems for Cybersecurity: Design Principles and Implementation Challenges,” *ACM Transactions on Information and System Security*, vol. 26, no. 3, pp. 1–29, 2023.

[25] P. K. Taylor, S. R. Johnson, and A. M. Lee, “Distributed Validation Mechanisms for Collaborative Security Research Platforms,” *IEEE Computer*, vol. 57, no. 4, pp. 78–87, 2024.

[26] L. K. Anderson, T. S. Brown, and R. J. White, “Automated Discovery and Classification of Attack Patterns in Web Applications,” *Computers & Security*, vol. 135, pp. 89–105, 2024.

[27] Y. C. Chen, F. L. Wang, and J. M. Zhang, “Machine Learning Approaches for Intelligent Security Testing: A Comprehensive Survey,” *ACM Computing Surveys*, vol. 55, no. 11, pp. 1–45, 2023.

[28] R. P. Garcia, K. L. Davis, and S. M. Wilson, “Scalability Solutions for Blockchain-Based Security Applications,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 35, no. 6, pp. 1456–1471, 2024.

[29] M. S. Johnson, D. R. Miller, and A. T. Brown, “Contemporary Evasion Techniques in Web Application Security: Analysis and Countermeasures,” *Journal of Information Security and Applications*, vol. 74, pp. 103–456, 2023.

[30] H. W. Lee, P. M. Thompson, and C. L. Anderson, “Security Analysis of Consensus Mechanisms in Permissioned Blockchain Networks,” *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 2567–2581, 2024.

[31] S. P. Wilson, R. K. Chen, and M. J. Davis, “Automated Testing Frameworks for Modern Web Application Security,” *Software Testing, Verification and Reliability*, vol. 33, no. 4, pp. 245–267, 2023.

[32] C. A. Martinez, L. S. Taylor, and D. M. Johnson, “AI-Driven Security Testing: Challenges and Opportunities in the Modern Threat Landscape,” *AI Magazine*, vol. 45, no. 2, pp. 123–139, 2024.

[33] T. L. Brown, K. R. Anderson, and J. P. Wilson, “Cryptographic Verification Mechanisms for Distributed Security Systems,” *Journal of Cryptology*, vol. 36, no. 4, pp. 234–256, 2023.

[34] E. M. Rodriguez, A. S. Kim, and P. T. Lee, “Collaborative Research Platforms in Cybersecurity: Design Patterns and Implementation Strategies,” *IEEE Security & Privacy*, vol. 22, no. 3, pp. 45–54, 2024.