Generating Evasive Payloads for Assessing Web Application Firewalls with Reinforcement Learning and Pre-trained Language Models

DOI: https://doi.org/10.54654/isj.v2i25.1128

Tran Gia Bao, Dinh Cong Duc, Phan The Duy*

***Abstract*— Web Application Firewalls (WAFs) serve as a critical defense mechanism against various web-based attacks such as SQL Injection (SQLi), Cross-Site Scripting (XSS), Server-Side Request Forgery (SSRF), Remote Code Execution (RCE), and NoSQL Injection. However, modern adversaries often craft evasive and obfuscated payloads capable of bypassing traditional WAF rules. To effectively assess and challenge the robustness of WAFs, we propose DEG-WAF, a Deep Evasion Generation framework that leverages Large Language Models (LLM) in conjunction with Reinforcement Learning (RL) to generate evasive payloads against WAFs. The system consists of four core components: a payload generation agent based on a pre-trained LLM (OPT-125M), a reward model that approximates WAF behavior, a grammar-based sampling agent that ensures syntactic validity, and an RL agent trained with either Proximal Policy Optimization (PPO) or Advantage Actor-Critic (A2C) to fine-tune generation strategies. Experimental evaluations on real-world WAFs, including ModSecurity and SafeLine, demonstrate that the A2C-based model significantly outperforms baseline LLMs—achieving a bypass success rate of 80.16% on SQLi and 74.70% on NoSQLi for ModSecurity, and 97.8% on RCE for SafeLine. These results underscore the potential of our LLM-RL framework to serve as a robust foundation for evaluating and enhancing the resilience of WAF systems under adversarial conditions.**

***Tóm tắt*— Tường lửa ứng dụng Web (WAF) đóng vai trò là một cơ chế phòng thủ quan trọng chống lại nhiều dạng tấn công dựa trên web như SQL Injection (SQLi), Cross-Site Scripting (XSS), Server-Side Request Forgery (SSRF), Remote Code Execution (RCE) và NoSQL Injection. Tuy nhiên, các tin tặc hiện đại, tinh vi thường tạo ra các payload được làm rối và né tránh nhằm vượt qua các luật WAF truyền thống. Để đánh giá và thách thức hiệu quả độ bền vững của WAF, Để đánh giá và thách thức hiệu quả độ mạnh mẽ của các Tường lửa ứng dụng web (WAF), nhóm tắc giả đề xuất DEG-WAF, một khung công tác Tạo sinh Payload né tránh sâu (Deep Evasion Generation) tận dụng mô hình ngôn ngữ Lớn (LLM) kết hợp với học tăng cường (RL) để tạo ra các payload né tránh nhằm chống lại các WAF. Hệ thống bao gồm bốn thành phần chính: một tác tử sinh payload dựa trên LLM đã được huấn luyện trước (OPT-125M), một mô hình thưởng mô phỏng hành vi của WAF, một tác tử lấy mẫu dựa trên ngữ pháp để đảm bảo tính hợp lệ cú pháp, và một tác tử RL được huấn luyện với thuật toán Proximal Policy Optimization (PPO) hoặc Advantage Actor-Critic (A2C) nhằm tinh chỉnh chiến lược sinh. Các đánh giá thực nghiệm trên các WAF thực tế, bao gồm ModSecurity và SafeLine, cho thấy mô hình dựa trên A2C vượt trội đáng kể so với các LLM nguyên bản — đạt tỷ lệ né tránh 80.16% với SQLi và 74.70% với NoSQLi trên ModSecurity, và 97.8% với RCE trên SafeLine. Những kết quả này nhấn mạnh tiềm năng của khung LLM-RL do chúng tôi đề xuất như một nền tảng vững chắc để đánh giá và nâng cao khả năng chống chịu của các hệ thống WAF trong điều kiện đối kháng.**

Keywords*— *Web Application Firewall, reinforcement learning, large language model, payload generation, grammar attacks.

Từ khóa*— *Tường lửa ứng dụng web; học tăng cường; ngôn ngữ lớn; tác tử sinh payload.

I. Introduction

Web applications are increasingly exposed to sophisticated cyberattacks [1], including SQL Injection (SQLi), Cross-Site Scripting (XSS),

This manuscript was received on July 31, 2025. It was reviewed September 15, 2025, revised on September 23, 2025 and accepted on September 29, 2025.

* Corresponding author.

Server-Side Request Forgery (SSRF), and NoSQL Injection. To mitigate such threats, Web Application Firewalls (WAFs) are widely adopted as a frontline defense mechanism, operating by filtering HTTP traffic based on predefined rules or anomaly-based detection models [2]. Despite their widespread deployment, traditional WAFs often rely on static, manually crafted rules that are slow to adapt to evolving obfuscation and evasion techniques [3, 4]. Consequently, adversaries can increasingly construct semantically equivalent yet syntactically obfuscated payloads that successfully bypass these protections.

Ensuring the robustness of WAFs under adversarial conditions has therefore become a critical research objective [4 – 7]. While conventional testing approaches-such as white-box analysis and mutation-based fuzzing—have achieved limited success, they frequently suffer from constraints in payload diversity, scalability, and adaptability. Additionally, Reinforcement Learning (RL)-based search techniques applied over clustered payloads have shown potential for deeper exploration, but often fall into local optima and struggle with generalization across WAF architectures.

Recent advancements in machine learning (ML) have introduced two enabling paradigms for addressing these limitations. First, Large Language Models (LLMs) have demonstrated remarkable capabilities in structured text generation, adversarial input crafting, and semantic generalization across domains [8, 9]. While these models are often perceived as primarily beneficial for long text processing, their strength in fact lies in modeling structured token sequences of any length. Web attack payloads, although short in form, demand strict syntactic validity (e.g., quotation marks, logical operators, or encoding rules) and semantic coherence to remain functional [10]. By leveraging large-scale pre-training and the autoregressive Transformer architecture, LLMs can generate payloads that are both syntactically correct and diverse in their variants [11 – 13]. This capability positions LLMs as more suitable than traditional fuzzing or rule-based methods, which often fail to simultaneously ensure correctness and diversity in generating short adversarial inputs. Second, RL provides a sequential decision-making under uncertainty, and has been increasingly applied to security-related tasks where adaptive exploration is critical [14 – 16].

Motivated by these developments, we propose a novel WAF evaluation framework that integrates pre-trained LLMs with RL to autonomously generate evasive web attack payloads in a black-box setting. Our architecture consists of three primary components: (i) a token-level payload generator built upon a pre-trained LLM (OPT-125M), (ii) a reward model trained to approximate the decision boundary of the target WAF, and (iii) a policy optimization agent - based on either Proximal Policy Optimization (PPO) or Advantage Actor-Critic (A2C)-that fine-tunes the LLM to generate increasingly evasive samples. The reward model serves as a learned surrogate for the WAF, providing dense, differentiable feedback signals that enable efficient RL training without requiring continuous queries to the actual WAF. To ensure structural correctness and domain relevance, we further incorporate a grammar-guided sampling mechanism that constrains generation to syntactically valid, attack-specific patterns.

We evaluate the effectiveness of our proposed framework against two representative WAF systems, including ModSecurity [17], a widely-used rule-based open-source firewall, and SafeLine [18], a commercial WAF employing machine learning-based behavioral detection. Our evaluation spans five major attack classes-SQLi, NoSQLi, XSS, SSRF, and RCE - demonstrating the generality and robustness of the approach across both rule-based and ML-based WAFs.

The main contributions of this paper are as follows:

- We introduce DEG-WAF, a robustness evaluation framework that integrates a pre-trained Large Language Model (OPT-125M) with RL algorithms (PPO/A2C) to automatically generate evasive web attack payloads. The proposed approach enables adaptive learning based solely on WAF responses, in contrast to traditional mutation-based or static fuzzing strategies.
- We develop a reward model trained to approximate the decision boundary of the target WAF, thereby allowing scalable and

differentiable RL in black-box settings without requiring direct queries to the WAF during training.
- We incorporate a grammar-guided sampling mechanism to constrain generation to syntactically valid and semantically meaningful attack payloads, improving the structural integrity and attack specificity of the generated inputs.
- We conduct comprehensive empirical evaluations on two representative WAFs-ModSecurity (signature-based) and SafeLine (machine learning-enhanced)-across five major attack types: SQLi, NoSQLi, XSS, SSRF, and RCE. The results demonstrate that our framework significantly improves bypass success rates while maintaining generalization across heterogeneous WAF architectures.

The remainder of this paper is organized as follows. Section II introduces the foundational background, including WAFs, LLMs, and RL. In Section III, we present the threat model underlying this study, which characterizes the behavior of adversaries attempting to evade WAFs. Section IV presents our proposed framework, emphasizing the role of the reward model as a learned WAF surrogate and training environment. In Section V, we give the details of our experimental setup and evaluation metrics. In Section VI, the results of our framework on real-world WAFs are presented and analyzed. Section VII provides a discussion on the limitations of our approach and outlines directions for future improvements. Finally, Section VIII concludes the paper.

II. Background and Related Works

A. Web Application Firewalls

Web Application Firewalls (WAFs) [2] are security mechanisms that inspect and filter HTTP traffic between clients and web applications, aiming to detect and block malicious inputs such as SQL Injection (SQLi), Cross-Site Scripting (XSS), Server-Side Request Forgery (SSRF), and related attack vectors. WAFs typically operate using either predefined rule sets or anomaly detection models.

Two primary categories of WAFs are commonly adopted: rule-based and machine learning (ML)-based. Rule-based WAFs, such as ModSecurity [17] and Naxsi [19], rely on manually crafted signature rules (e.g., OWASP Core Rule Set [20]) to identify known attack patterns. While effective against well-understood threats, these systems are often susceptible to obfuscated or previously unseen payloads. In contrast, ML-based WAFs [21] attempt to learn generalized attack behaviors from data, offering potential adaptability but facing challenges such as reduced interpretability and higher false positive rates.

Assessing the resilience of WAFs is essential for ensuring their reliability in adversarial settings. Black-box testing - where the internal logic of the WAF is unknown - has emerged as a practical evaluation method [5, 6]. However, it necessitates the development of advanced payload generation strategies capable of systematically probing the WAF's decision boundaries.

B. Large Language Models for Adversarial Payload Generation

Large Language Models (LLMs), such as GPT [22, 23], OPT [24], or LLaMA [25], are transformer-based architectures trained on massive text corpora [26]. These models have demonstrated remarkable comprehension, reasoning, and generalization capabilities across a wide range of domains, making them a cornerstone in advancing artificial intelligence applications. Through pre-training on large-scale unlabeled data followed by task-specific fine-tuning, LLMs can perform diverse downstream tasks guided by human instructions, and are considered as promising enablers toward scalable automation and adaptable intelligence.

In the context of security applications, LLMs can be leveraged to generate adversarial inputs that mimic real-world attack payloads [27, 28]. Their ability to generalize from examples enables them to explore variations of known exploits and potentially discover novel evasion techniques. When guided by domain knowledge or optimization signals, LLMs [8] become powerful tools for testing and evaluating security mechanisms like Web Application Firewalls (WAFs).

Unlike traditional fuzzing techniques that rely on hand-crafted rules or shallow mutation operators, LLMs can model rich syntactic and

semantic patterns of attack payloads through pretraining and transfer learning. This allows them to construct structurally valid inputs that preserve functional intent while evading detection. Moreover, token-level autoregressive generation enables fine-grained control over the construction process, which is essential for tasks such as SQLi or XSS crafting where precise injection positioning matters.

When combined with search-based optimization methods - such as RL or grammar-guided sampling - LLMs can adaptively refine their generation strategies based on feedback signals. This adaptability makes them particularly effective in black-box testing settings, where internal logic of the target system is unknown and exploratory learning is required to identify evasive behaviors.

C. Reward Model for WAF Behavior

In RL-based systems for security testing, the environment typically provides feedback in the form of rewards, which guide the agent's learning process[29]. However, directly querying a real WAF for each generated payload is often inefficient, non-differentiable, and resource-intensive. To overcome these limitations, a *reward model* is commonly introduced to approximate the behavior of the WAF.

The reward model acts as a learned proxy that predicts the likelihood of a payload successfully bypassing the WAF. By replacing the WAF during training, it enables the RL agent to receive dense and informative feedback signals. This not only accelerates training by eliminating the need for constant WAF interaction, but also enables the use of gradient-based optimization methods.

In essence, the reward model serves as a differentiable and scalable environment for the agent, capturing the decision boundary of the target WAF based on previously observed behavior. It plays a crucial role in making RL practical and effective for adversarial payload generation.

D. Grammar-Guided Payload Sampling

Grammar-guided techniques have been widely adopted in fuzzing to generate structured inputs that adhere to predefined syntactic rules [30 –32]. Within the domain of WAF testing, attack grammars formalize the structure of injection payloads, thereby ensuring that generated inputs remain syntactically valid and domain-relevant.

By leveraging these grammar definitions, the input space can be systematically explored through guided sampling strategies. When integrated with learning signals - such as reward feedback from model predictions - the sampling process becomes adaptive, dynamically prioritizing grammar branches that yield higher probabilities of bypassing WAF detection. This hybrid approach effectively balances structural validity with exploratory diversity, enhancing both the syntactic correctness and evasive potential of generated payloads.

E. RL for Security Testing

RL [15] is a ML paradigm in which an agent learns to make sequential decisions by interacting with an environment and receiving feedback in the form of rewards. In the context of security testing, RL can be used to guide the generation of adversarial payloads that maximize the chance of bypassing security mechanisms such as WAFs.

A typical RL framework is formulated as a Markov Decision Process (MDP)[36], defined by:

- State: The current representation of the environment, such as the partial payload generated so far.
- Action: A decision the agent makes, such as selecting the next token to append.
- Environment: The system that evaluates actions, which in WAF testing can be approximated by a learned reward model.
- Reward: A numerical signal indicating the success of the action, e.g., the likelihood of bypassing a WAF.
- Policy: A strategy or model that maps states to actions to maximize cumulative reward.

RL algorithms can be categorized into three main types:

- Value-based methods: These approaches, such as Q-learning and Deep Q-Networks (DQN)[37], estimate the value of actions and select the one with the highest expected reward.
- Policy-based methods: These directly learn the policy without estimating value functions. Algorithms such as REINFORCE

TABLE I. COMPARATIVE ANALYSIS OF RELATED WORKS

Work	RL	LLM	Grammar	Attack Types	WAF Targets
GPTFuzzer[6]	Yes	GPT-2	Yes	SQLi, XSS, RCE	ModSecurity, Naxsi
AdvSQLi[33]	No	No	Yes	SQLi	Real-world WAFaaS
DaNuoYi[34]	No	No	Partial	SQLi, XSS, XMLi, OSi, PHPi	ModSecurity, Ngx-WAFs
WAFBooster[35]	No	No	No	SQLi, XSS, CMDi	8 Commercial WAFs
DEG-WAF (Ours)	**Yes**	**OPT-125M**	**Yes**	**SQLi, NoSQLi, XSS, SSRF, RCE**	**ModSecurity, SafeLine**

and Proximal Policy Optimization (PPO)[38] fall into this category.

- Actor-Critic methods: These combine both value estimation and policy learning. A notable example is Advantage Actor-Critic (A2C)[39], which uses an actor to update the policy and a critic to evaluate it.

In security testing scenarios, policy-based and actor-critic methods are often favored due to their effectiveness in handling large, continuous action spaces such as token-level text generation. These approaches allow agents to adaptively explore the payload space and discover evasive strategies that static fuzzing methods may overlook.

F. Comparative Analysis with Related Works

Recent research on adversarial testing of WAFs has followed two primary directions: (i) approaches leveraging reinforcement learning (RL) and generative models, and (ii) mutation-based or heuristic strategies without learning mechanisms. Table I provides a structured comparison between our framework and four representative studies: GPTFuzzer [6], AdvSQLi [33], DaNuoYi [34], and WAFBooster [35].

In the generative category, GPTFuzzer [6] represents an early attempt to combine RL with large language models by fine-tuning GPT-2 using Proximal Policy Optimization (PPO) to produce evasive payloads for SQLi, XSS, and RCE. While it introduces grammar-aware generation and supports multiple attack types, its lack of a reward model abstraction limits scalability and adaptability in black-box testing environments.

AdvSQLi [33], on the other hand, employs a mutation-based adversarial strategy guided by grammar rules to test WAF-as-a-Service platforms. Although it demonstrates strong effectiveness against commercial WAFs, its focus is limited to SQLi and does not involve learning-based adaptation, reducing its applicability to more complex or previously unseen attack scenarios.

DaNuoYi [34] expands the scope by proposing a multi-task evolutionary testing framework covering a wider range of attack types, including XSS, XMLi, and PHPi. It partially utilizes grammar guidance and incorporates environmental feedback for fitness evaluation. However, the absence of LLMs restricts semantic generation quality, and evolutionary algorithms may suffer from convergence issues and syntactic instability.

Shifting from offense to defense, WAFBooster [35] focuses on improving WAF rule sets by generating evasive payloads through shadow models and RNN-based generators. While effective for rule hardening, it does not address the need for comprehensive and adaptive payload generation for evaluation purposes.

In contrast to the above, our work integrates the strengths of prior approaches while addressing their limitations. We utilize an OPT-125M LLM fine-tuned via RL (PPO or A2C) in a black-box setting, guided by a reward model that approximates WAF behavior to enable efficient learning. Additionally, grammar-guided generation ensures syntactic validity and attack specificity. Unlike prior works, our framework supports a broader spectrum of attack types—SQLi, NoSQLi, XSS, SSRF, and RCE—and is validated against both rule-based

(ModSecurity) and ML-based (SafeLine) WAFs.

This comprehensive design addresses key limitations in scalability, semantic richness, and multi-vector support, offering an adaptive and extensible solution for WAF robustness evaluation across heterogeneous defense paradigms.

III. Threat Model

A. Problem Formulation and Threat Model

Our research addresses the challenge of thoroughly evaluating WAF effectiveness against dynamic and polymorphic web attacks. Traditional WAF testing approaches, which often rely on static signatures or replaying known attack samples, are insufficient to capture sophisticated evasive behaviors commonly observed in modern adversarial payloads. To overcome this limitation, we propose an AI-driven framework that dynamically generates novel, semantically valid payloads capable of bypassing WAFs.

Threat Model. We adopt a black-box adversarial setting, where the attacker has no knowledge of WAF rules, configurations, or the internal logic of the protected web application. The attacker can only observe the WAF's binary decision outcomes, such as HTTP response codes (e.g., 200 OK for allowed requests and 403 Forbidden for blocked ones). This threat model realistically simulates an external adversary attempting to probe and adaptively bypass deployed WAFs without insider access.

The attack surface in this scenario includes all publicly accessible web interfaces protected by the WAF, especially HTTP GET/POST parameters and headers vulnerable to injection-based attacks. The adversary is assumed to have the ability to send arbitrary HTTP requests and receive server responses, but cannot exploit out-of-band channels or perform authenticated requests unless credentials are publicly known.

Our system focuses on generating attack payloads across multiple vectors—including SQLi, RCE, XSS, SSRF, and NoSQLi—while ensuring that the payloads remain functional and malicious in intent. These payloads are designed to bypass WAF filtering mechanisms and reach the backend server, simulating realistic exploitation attempts.

This black-box adversarial setup is consistent with prior work on evasion attacks against security systems. For instance, the WAF-A-MoLE framework [5] explores evading WAFs in a black-box setting by using substitute models and adversarial perturbations to craft inputs that bypass detection. Similarly, the GPTFuzzer framework [6] leverages pre-trained LLMs to generate adaptive adversarial payloads in a black-box setting to evaluate WAF robustness. Inspired by such approaches, our framework leverages RL and grammar constraints to explore the attack space more efficiently while preserving functional semantics of the payload.

Objective: Under this threat model, the core objective is to craft web attack payloads that:

- Successfully bypass WAF detection mechanisms (i.e., yield HTTP 200 responses),
- Preserve malicious functionality relevant to the target class of vulnerability,
- Demonstrate generalization to unseen WAF configurations,
- Exhibit diversity and novelty to avoid detection by signature-based methods.

Our framework is evaluated against both rule-based (ModSecurity with OWASP CRS) and machine learning-based (SafeLine) WAFs to highlight its generalizability across different defense paradigms.

B. Data Collection and Preprocessing

To enable effective training of the proposed LLM-based and RL-based components, a comprehensive and structurally diverse dataset of web attack payloads is constructed using grammar-guided synthesis. The grammar definitions are manually derived from commonly observed patterns across five major attack types: SQL Injection (SQLi), Cross-Site Scripting (XSS), Server-Side Request Forgery (SSRF), Remote Code Execution (RCE), and NoSQL Injection. These grammars are curated based on publicly available knowledge bases, including PayloadAllTheThings [40] and Hacktricks [41], ensuring both practical relevance and coverage of real-world exploitation techniques.

Payload instances are programmatically generated from the defined grammars to maximize syntactic variability while preserving

semantic correctness. Prior to model ingestion, each payload is normalized and tokenized into subword units compatible with Transformer-based architectures. This tokenization strategy preserves essential characters, operators, and encoding structures critical to the functional semantics of the attacks. All tokenized sequences are padded or truncated to a fixed maximum length, and each token is mapped to a unique integer identifier using a predefined vocabulary. This preprocessing pipeline ensures uniform input representation across the LLM, reward model, and RL agent.

IV. Methodology

This section details the DEG-WAF framework for assessing Web Application Firewalls using RL and pre-trained Large Language Models. Our system autonomously generates diverse and evasive attack payloads, mimicking an intelligent adversary to evaluate WAF robustness against SQLi, XSS, SSRF, RCE, and NoSQLi.

A. Overall System Architecture

Our proposed DEG-WAF system is a multi-component, closed-loop adaptive framework designed for continuous learning and optimization, as illustrated in Figure 1. The architecture integrates four primary interacting agents: the LLM Agent (Payload Generator), the Reward Agent (Payload Quality Evaluation Model), the Grammar Sampling Agent (Oriented Payload Generation), and the RL Agent (Policy Fine-tuning). These agents operate in an iterative cycle: the RL Agent prompts the LLM Agent to generate a payload, which the Reward Agent then evaluates to inform the RL Agent's policy updates, driving the system towards more effective payload generation.

B. Details of Architectural Module

Our DEG-WAF framework's core comprises the development and training methodologies for its specialized AI models.

1. LLM Agent (Payload Generator)

The LLM forms the foundation for generating diverse and nuanced attack payloads.

- Model Selection: We utilize OPT-125m from Meta AI, a 125-million-parameter Transformer decoder model.
- Pre-training Objective: The LLM is pre-trained on the combined dataset of synthesized attack payloads for all targeted types. The objective is to maximize the likelihood of generating these sequences, thereby teaching the model the syntax, semantics, and common obfuscation patterns of web attacks. The training objective is:

$$\mathcal{L}_{\text{LM}}(\theta) = -\sum_{i=1}^{N} \log P_\theta(x_i)$$

Standard training techniques are applied.

2. Reward Agent (Payload Quality Evaluation Model)

This agent provides a continuous, differentiable signal to the RL Agent, simulating the WAF's response.

- Architecture: The Reward Agent uses the OPT-125 backbone with an added Linear Head to predict a bypass probability $\hat{r} \in [0, 1]$.
- Training Objective: The model is trained using a binary cross-entropy loss function to distinguish between WAF-bypassing and WAF-blocking payloads:

$$\mathcal{L}_{\text{reward}} = -\sum_{i=1}^{N} [y_i \log \hat{r}_i + (1 - y_i) \log(1 - \hat{r}_i)]$$

where $y_i \in \{0, 1\}$ is the true label.

3. Grammar Sampling Agent (Oriented Payload Generation)

The Grammar Sampling Agent is designed to generate syntactically valid and attack-specific payloads by leveraging a set of formal grammars $\mathcal{G} = \{G_1, G_2, ..., G_K\}$, where each grammar G_k corresponds to a particular vulnerability class (e.g., SQLi, XSS). Given a selected grammar G_k, the agent samples from the set of production rules $\mathcal{R}_k \subset G_k$, maintaining structural correctness and semantic intent throughout the generation process.

At each iteration, the sampling probability for a rule $r \in \mathcal{R}_k$ is defined as $P(r_t) = \frac{e^{\alpha Q(r_t)}}{\sum_{r' \in \mathcal{R}_k} e^{\alpha Q(r')}}$, where $Q(r_t)$ is the estimated quality (e.g., expected reward) of rule r_t and α is a temperature parameter controlling exploration. The quality signal $Q(r_t)$ is iteratively updated using feedback from the Reward Agent, enabling the sampler to adaptively focus on rules that are empirically more effective

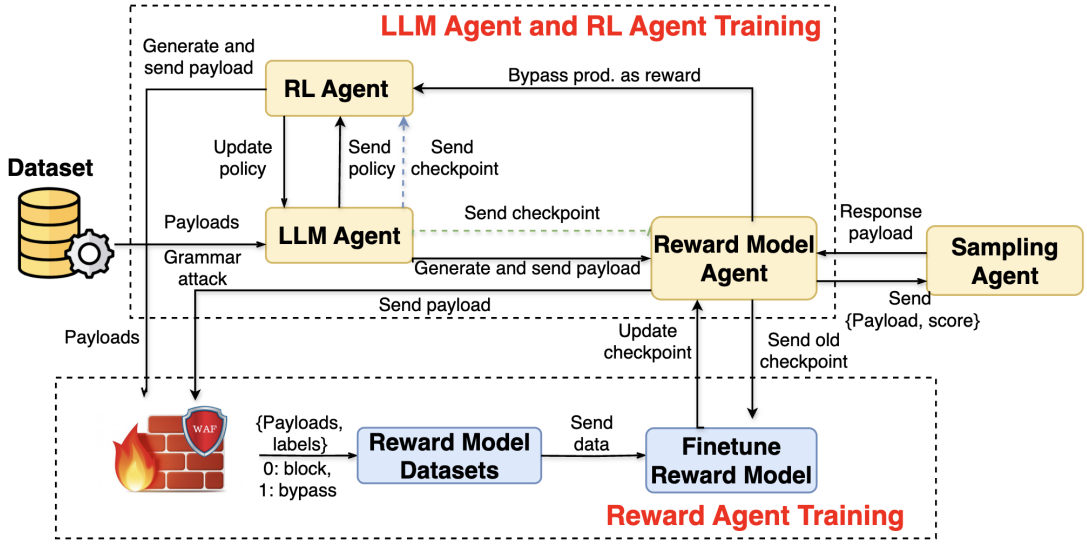


Figure 1. Overview of the proposed framework - DEG-WAF

at producing evasive payloads. Over time, this policy converges toward sampling distributions that emphasize both grammatical fidelity and evasive utility.

This probabilistic, feedback-driven mechanism bridges structured generation with reinforcement signals, improving the quality, validity, and evasiveness of the resulting adversarial samples.

4. RL Agent (Policy Fine-tuning)

Payload generation is formalized as a Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, P, R, \pi)$, where the state $s_t \in \mathcal{S}$ is the partially generated payload at step t, the action $a_t \in \mathcal{A}$ corresponds to the next token from the vocabulary, and the reward function $R(s_t, a_t)$ is provided by the learned reward model estimating bypass probability. The transition model P is implicitly determined by the LLM's autoregressive architecture, and the policy $\pi_\theta(a_t|s_t)$ is parameterized by the LLM.

To optimize this policy, we adopt Proximal Policy Optimization (PPO-Clip) [38], which maximizes the clipped objective:

$$\mathcal{L}^{\text{PPO}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right]$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio and $\hat{A}_t$ is the advantage estimate.

Separately, we also fine-tune the model using the Advantage Actor-Critic (A2C) algorithm [39] to evaluate its effectiveness relative to PPO. In A2C, the actor generates actions while the critic estimates the value function $V(s_t)$ to reduce gradient variance:

$$\hat{A}_t = R_t - V(s_t).$$

Entropy regularization $\beta \cdot \mathcal{H}[\pi(\cdot|s_t)]$ is added to promote exploration, and a KL-divergence term $\lambda \cdot \text{KL}[\pi_\theta || \pi_{\text{init}}]$ penalizes excessive deviation from the pre-trained LLM, preserving syntactic priors.

Each of these algorithms is applied independently to fine-tune the LLM policy, allowing a comparative evaluation of their respective effectiveness in guiding evasive payload generation.

V. Experimental Settings and Evaluation Metrics

A. Research Questions

In order to analyze the effectiveness of our proposed DEG-WAF framework, we identify a set of guiding research questions (RQs):

- RQ1: Can RL improve LLM-based payload generation against WAFs?
- RQ2: How does the performance vary across WAFs with different defense mechanisms?
- RQ3: What is the impact of grammar-guided sampling on RL-based payload generation?
- RQ4: How does our model perform in bypassing WAFs and validating attack payloads compared to other machine learning and deep learning algorithms?
- RQ5: How is the quality of the generated payloads quantitatively evaluated, particularly

concerning redundancy and component distribution?

B. Dataset and Experimental Setup

1. Data Generation and Characteristics

The effectiveness of LLMs and RL agents in generating evasive payloads is heavily influenced by the quality and representativeness of the training data. In this study, a grammar-driven dataset is constructed to ensure that the generated payloads are both syntactically correct and semantically aligned with specific attack vectors, including SQL Injection (SQLI), NoSQL Injection, Server-Side Request Forgery (SSRF), Cross-Site Scripting (XSS), and Remote Code Execution (RCE). The grammars are manually designed based on widely adopted repositories such as PayloadAllTheThings [40] and Hacktricks [41], providing comprehensive and practical coverage of real-world attack patterns. From these grammar rules, a total of 75,000 payloads are synthesized, with an equal distribution of 15,000 samples per attack type, ensuring data balance and diversity.

2. Training Configurations of DEG-WAF framework

Our training pipeline is structured around three core agents: a pre-trained Large Language Model (LLM) serving as the policy generator, a reward model acting as the WAF simulator, and a Reinforcement Learning (RL) agent that optimizes the policy. A final grammar-based agent refines low-quality payloads to ensure robustness and syntactic correctness.

a) LLM Fine-Tuning.

Each LLM is first fine-tuned on a dataset of 15,000 payloads specific to a particular attack type (e.g., SQLi, XSS, RCE). This supervised learning phase enables the model to internalize the nuanced syntax, obfuscation patterns, and domain-specific characteristics of the target attack vector. Fine-tuning is conducted over 4 epochs, using a learning rate of 2×10^{-5} and a batch size of 16. The resulting model serves as the policy backbone for subsequent reinforcement learning.

b) Reward Model Training.

To simulate Web Application Firewall (WAF) behavior, we train a dedicated reward agent to provide continuous and differentiable feedback. This agent is built upon the OPT-125m transformer backbone and is trained on the same set of 15,000 payloads, labeled according to real WAF verdicts. The training process spans 4 epochs and enables the model to learn a fine-grained scoring function that reflects WAF bypass success, which is used to guide the RL optimization process.

c) Reinforcement Learning Optimization.

Using the pre-trained LLM as the policy model, the RL agent generates an additional 10,000 payloads per attack type. These samples are iteratively optimized through either Proximal Policy Optimization (PPO) or Advantage Actor-Critic (A2C). The policy is updated based on reward signals returned by the reward model over 3 major training iterations, promoting the generation of more effective and evasive payloads.

- PPO Configuration:
  - Learning rate: 2×10^{-5}
  - PPO epochs: 4
  - Batch size: 16
  - Clip range: 0.2
  - Discount factor (γ): 0.9
  - Value loss coefficient: 0.5
  - Entropy coefficient: 0.01
  - Max gradient norm: 1.0
  - Max input length: 128 tokens
- A2C Configuration:
  - Learning rate: 1×10^{-5}
  - Training iterations: 3
  - Batch size: 16
  - Value loss coefficient: 0.5
  - Entropy coefficient: 0.01
  - Clip range (for stability): 0.2
  - Max gradient norm: 1.0
  - Max input length: 128 tokens

d) Grammar-Based Refinement Agent.

As a final post-processing step, a grammar-based agent is applied to refine any payloads with a reward score below a set threshold (e.g., 0.8). This agent uses a formalized attack grammar and a weighted-random expansion strategy to reconstruct payloads, correcting structural flaws and improving overall quality before inclusion in the final output set.

3. Configuration of Deep Learning Baseline Models

The following deep-learning baselines are used as comparative generators: we evaluate each model's ability to (i) evade web application firewalls (reported as *Bypass %*) and (ii) produce functional exploits among the bypassed payloads (reported as *Validate %*).

a) Overview.

All deep-learning baselines were trained and evaluated under a common experimental protocol to ensure an apples-to-apples comparison with the proposed model. Training data, preprocessing, and splits are identical across methods: we use the same cleaned corpus of attack payloads (grammar-derived samples, mutated examples, and prior LLM outputs) with a same dataset. Each model generates 5,000 payloads per attack type during evaluation. Where applicable, pretrained checkpoints were fine-tuned on the training split. To ensure reproducibility, experiments were run with fixed random seeds (three different seeds for each reported result) and we report mean values. Sampling/decoding hyperparameters were held consistent across models (unless a specific model class requires different decoding, which is noted below).

b) Common training decoding settings.

- Optimizer: AdamW (or Adam for RNNs).
- Learning-rate schedule: constant LR with optional linear warmup where appropriate.
- Weight decay: 0.01 for transformer-based models; 0 for RNN baselines.
- Batch size: 16–64 depending on model memory footprint (reported per model below).
- Early stopping on validation loss with patience = 3 epochs.
- Tokenization: byte-level or subword tokenizer (same tokenizer for all transformer baselines).
- Decoding (generation): sampling with `top-p`=0.95 and temperature = 0.8 for stochastic samplers; beam search (beam size = 5) used for deterministic Seq2Seq baselines where appropriate. All models produced up to a fixed maximum sequence length per payload (128 tokens) or until a task-specific delimiter.

c) LSTM (autoregressive RNN baseline)

The LSTM baseline is implemented as a character-level autoregressive language model trained to predict the next character in a payload sequence. Input payloads are concatenated into a single text stream and tokenized at the character level using a dataset that builds a sorted character vocabulary; sequences of length 128 characters are extracted with one-step-ahead targets.

- Model: char-level 2-layer LSTM (embedding=256, hidden=256).
- Tokenization: character-level; sequence length = 128.
- Training: Adam, lr=1e-3; loss = CrossEntropy; batch=64; epochs=10.
- Regularization: no dropout in provided implementation.
- Generation: use ancestral sampling; max_len=100; temperature (exposed) default=1.0 (recommend 0.8 for reported runs).
- Evaluation budget: 5,000 generated payloads per attack type.

d) Seq2Seq (encoder–decoder) baseline

A character-level encoder–decoder Seq2Seq model is used as a generative baseline. The model and training follow the provided implementation:

- Tokenization & dataset: char-level tokenizer with special tokens (`<PAD>`, `<SOS>`, `<EOS>`, `<UNK>`); generator-only dataset with source = single `<SOS>` token and target payloads padded/truncated to `max_tgt_len`=120.
- Encoder: bidirectional LSTM, embedding size = 64, encoder hidden size = 128, num_layers = 1 (bidirectional states combined by summation).
- Decoder: unidirectional LSTM, embedding size = 64, decoder hidden size = 128, num_layers = 1, linear output projection to vocabulary.
- Training: Adam optimizer, lr = 1×10^{-3}; CrossEntropyLoss with `ignore_index` set to pad token; gradient clipping = 1.0. Reference run: batch size = 16, epochs = 30, teacher_forcing_ratio = 0.5.
- Generation: stepwise sampling from decoder logits using temperature and optional top-k filtering (function supports `temperature`

and `top_k`); maximum generation length = 120; deterministic behavior achieved when `temperature` ≤ 0.

e) Flan-T5-small (pretrained Transformer fine-tuned)

We fine-tuned the public google/flan-t5-small encoder–decoder checkpoint on payload generation using the HuggingFace `Trainer` API.

- Base checkpoint: google/flan-t5-small (fine-tuned on the task-specific payload corpus).
- Tokenizer & prompt: model tokenizer is used unchanged; training examples are created with a short instruction prefix and targets padded/truncated to `max_length`=128.
- Training driver & hyperparameters: The Trainer uses the library's default AdamW optimizer unless overridden.
- Generation / sampling: stochastic decoding is used in evaluation with `do_sample=True`, `top_p`=0.95 and `top_k`=50 (max_length configurable, e.g. 100). Deterministic decoding (greedy/beam) can be used for ablation.

f) SeqGAN (adversarial generator)

SeqGAN is implemented for character-level sequence generation following standard practices in adversarial text modeling.

- Generator: LSTM-based generator, pretrained with negative log-likelihood (NLL) loss for stability.
- Discriminator: CNN-based binary classifier operating on token sequences.
- Training: two-stage — (1) MLE pretraining of the generator, (2) adversarial training using REINFORCE with discriminator-provided rewards.
- Optimizer / LR: Adam optimizer with a learning rate of $1e{-}4$ for both generator and discriminator; gradient clipping applied.
- Decoding: sampling from the generator's policy with temperature $\tau = 0.8$.

g) GPTFuzzer (autoregressive Transformer + RL)

GPTFuzzer is a decoder-only, GPT-style generative baseline that is adapted with reinforcement learning to maximise WAF-bypassing payloads. In practice we follow a pretrain–reward–RL pipeline: a small autoregressive Transformer is pretrained/fine-tuned on payload corpora, a lightweight reward model is trained on WAF labels to predict bypass probability, and the policy is further adapted with PPO while applying a KL penalty to the pretrained model to limit policy drift.

- Architecture: decoder-only Transformer (GPT-style) used as the generation policy.
- Reward model: separate classifier/regressor trained on WAF outcomes to provide dense bypass rewards.
- RL fine-tuning: Proximal Policy Optimization (PPO) with a KL regularizer to prevent collapse; reward = predicted bypass probability.
- Generation: stochastic autoregressive sampling from the learned policy to produce diverse payloads for evaluation.
- Implementation notes: implemented with PyTorch / HuggingFace tooling; experiments averaged across runs for stability and checkpoints saved for reproducibility.

4. WAF Targets

The evaluation is conducted on two Web Application Firewalls with contrasting detection mechanisms. ModSecurity, a widely used open-source WAF, employs the OWASP Core Rule Set (CRS) and relies on signature-based detection, making it representative of traditional rule-based systems. In contrast, SafeLine is an open-source WAF that incorporates semantic analysis and ML techniques, targeting zero-day attack patterns. The inclusion of both WAF types enables a comprehensive assessment of the generalizability and adaptability of the proposed payload generation framework.

C. Evaluation Metrics

To provide a quantitative assessment of payload quality, we evaluate five key metrics, including Duplicate Rate, Diversity, Entropy, Bypass Rate, Validate Rate. Therein, experiments are conducted on 5,000 generated payloads for each attack type.

1. Duplicate Rate (Dupl)

This metric measures the proportion of identical payloads within a generated set. To compute this, each payload p in the set of all payloads P is first converted into a unique MD5

hash, $H(p)$. The duplicate rate is then defined as one minus the ratio of the number of unique hashes to the total number of payloads. A lower value indicates greater novelty.

$$\text{Duplicate Rate} = 1 - \frac{|\{H(p) \mid p \in P\}|}{|P|}$$

2. Diversity (Div)

This metric quantifies the lexical richness of the generated payloads using the Type-Token Ratio (TTR). All payloads in a set are first tokenized by splitting them into individual terms (tokens). The diversity is the ratio of the number of unique tokens (V, vocabulary size) to the total number of tokens (N). A higher TTR indicates a broader and more varied vocabulary.

$$\text{Diversity (TTR)} = \frac{V}{N}$$

3. Entropy (Ent)

This metric measures the uniformity and unpredictability of the token distribution, based on Shannon Entropy. It quantifies the average amount of information, or "surprise," associated with each token. For a vocabulary of size V, the entropy $H(X)$ is calculated based on the probability $p(x_i)$ of each unique token x_i appearing in the set. A higher entropy value signifies a more balanced and less predictable distribution of attack components.

$$\text{Entropy } H(X) = -\sum_{i=1}^{V} p(x_i) \log_2 p(x_i)$$

4. Bypass Rate:

Bypass measures a generator's ability to evade a target WAF: specifically, the share of generated payloads that the WAF does not block. We compute Bypass as the unconditional proportion of generated payloads that were observed to pass the WAF.

$$\text{Bypass } (\%) \;=\; 100 \times \frac{N_{\text{bypassed}}}{N_{\text{generated}}}$$

Where:

- $N_{\text{generated}}$: total number of payloads generated for the given attack type (in our experiments $N_{\text{generated}} = 5{,}000$).
- N_{bypassed}: number of generated payloads that successfully evaded the WAF (i.e., were not blocked).

5. Validate Rate:

Validate measures the functional effectiveness of payloads conditional on evasion: the proportion of bypassed payloads that produce the intended server-side effect (i.e., a true exploit). We compute Validate over the bypassed subset as:

$$\text{Validate } (\%) = 100 \times \frac{N_{\text{validated}}}{N_{\text{bypassed}}}.$$

Where the terms are defined as follows:

- N_{bypassed}: number of generated payloads that successfully evaded the WAF (i.e., were not blocked). This is counted regardless of whether the payload later produced any server-side effect.
- $N_{\text{validated}}$: number of generated payloads that both (a) bypassed the WAF *and* (b) met the strict validation criteria for the attack type (e.g., caused a DB error or data leakage for SQLi, rendered a DOM change for XSS, etc.).

VI. Evaluation Results

A. RQ1: Can RL improve LLM-based payload generation against WAFs?

Table II and Table III show the bypass rates on ModSecurity and SafeLine for each attack type. The RL-enhanced models (OPT-PPO and OPT-A2C) demonstrate improved or competitive performance compared to the base OPT-125m on both WAF systems.

On ModSecurity, OPT-A2C achieves the highest bypass rates on SQLI (80.16%) and NoSQLI (74.70%), while OPT-PPO leads on SSRF (65.52%) and XSS (7.86%). All models struggle to bypass RCE and XSS, highlighting ModSecurity's strong detection rules for these attacks.

On SafeLine, the bypass rates for RCE and SSRF are significantly higher across all models, with OPT-A2C achieving the highest RCE rate (97.80%) and OPT-PPO slightly ahead in SSRF (84.82%) and XSS (62.9%). This contrast indicates that SafeLine's ML-based detection is less robust against certain attack types, particularly payloads with structured grammar like RCE.

Across both WAFs, the RL models consistently outperform or match the base LLM, especially in SQLI, SSRF, and XSS. These results confirm that RL enhances the LLM's ability to generate evasive payloads and adapt to WAF behaviors.

TABLE II. *Evaluation of the effectiveness of 5000 payloads of various attack types generated from 3 models in bypassing WAF ModSecurity*

Model	SQLI	RCE	SSRF	NoSQLI	XSS
OPT-125m	74.56	20.40	64.88	70.81	6.90
OPT-PPO	74.78	**20.50**	**65.52**	73.46	**7.86**
OPT-A2C	**80.16**	20.44	60.90	**74.70**	7.84

TABLE III. *Evaluation of the effectiveness of 5000 payloads of various attack types generated from 3 models in bypassing WAF Safeline*

Model	SQLI	RCE	SSRF	NoSQLI	XSS
OPT-125m	92.04	94.96	82.10	N/A	48.70
OPT-PPO	86.04	99.70	82.18	N/A	48.58
OPT-A2C	**92.74**	**100**	**87.80**	N/A	**48.70**

B. RQ2: How does the performance vary across WAFs with different defense mechanisms?

To assess cross-WAF generalizability, we train the models on one WAF and evaluate their performance on another. Table IV presents the results when training on ModSecurity and testing on SafeLine, while Table V reports the inverse setup.

When trained on ModSecurity and tested on SafeLine, all models maintain high performance on SQLI and RCE, with OPT-PPO and OPT-A2C achieving a perfect 100% bypass rate on RCE. This suggests that RCE payloads learned on ModSecurity transfer well to SafeLine, likely due to weaker or less diverse RCE defenses in SafeLine. OPT-PPO also leads in SSRF (84.4%) and XSS (50.75%), indicating its RL policy enables better adaptability across WAF environments.

TABLE IV. *Effectiveness of 2000 payloads generated from models trained on ModSecurity and evaluated on SafeLine*

Model (Train on ModSecurity)	Test on SafeLine			
	SQLI	RCE	SSRF	XSS
OPT-125m	91.50	95.30	82.35	50.70
OPT-PPO	92.25	**100.00**	**84.40**	**50.75**
OPT-A2C	**93.65**	**100.00**	81.95	50.70

When trained on SafeLine and tested on ModSecurity, the performance drops significantly, especially for RCE. Although OPT-PPO and OPT-A2C previously achieved 100% bypass on SafeLine, they fall below 21% on ModSecurity, confirming that ModSecurity employs stricter filtering—particularly against high-impact payloads like RCE. Nonetheless, SQLI remains relatively stable for OPT-125m and OPT-A2C (above 74%).

TABLE V. *Effectiveness of 2000 payloads generated from models trained on SafeLine and evaluated on ModSecurity*

Model (Train on SafeLine)	Test on ModSecurity			
	SQLI	RCE	SSRF	XSS
OPT-125m	74.70	**20.60**	64.00	**7.75**
OPT-PPO	65.10	18.95	**72.95**	7.50
OPT-A2C	**74.80**	10.40	66.05	**7.75**

C. RQ3: What is the impact of grammar-guided sampling on RL-based payload generation?

To investigate the contribution of grammar-guided sampling, we compare two identical models: one using our Grammar Sampler and one without it. The Grammar Sampler constrains the generation process using domain-specific syntax rules for each attack type, such as SQLI or RCE. Table VI summarizes the bypass rates achieved by both configurations when tested on SafeLine WAF.

TABLE VI. *Effectiveness of 2000 payloads generated from two models (with and without Grammar Sampler) in bypassing WAF SafeLine*

Model	SQLI	RCE	SSRF	XSS
No Sampler	92.00	94.90	82.10	48.70
Sampler	**92.40**	**95.40**	**82.70**	**49.00**

The results in Table VI demonstrate that integrating the Grammar Sampler consistently improves bypass success rates across all four attack types. Although the improvements are relatively modest (ranging from 0.3% to 0.6%), they are steady and suggest that grammar-guided generation contributes positively to payload quality. Notably, the enhancements are most pronounced in SQLI and RCE, where accuracy and structural correctness are critical for successful evasion. In contrast, the gains in SSRF

and XSS, while smaller, are still meaningful - especially given the stricter detection rules applied to these attacks. These findings highlight the Grammar Sampler's ability to guide the generation process toward more valid and evasive attack patterns, making it a valuable component in the overall framework.

D. RQ4: How does our model perform in bypassing WAFs and validating attack payloads compared to other deep learning algorithms?

To validate that our generated payloads represent genuine attacks and not merely successful WAF bypasses, we test them against a custom web application within a controlled environment. A payload is considered valid only if it demonstrates a tangible, malicious impact post-evasion. The validation criteria are strict: a SQLi payload must exfiltrate data or induce a database error/delay; an XSS payload must render a new HTML tag in the DOM; an SSRF payload must be successfully parsed by at least one URL parser; a NoSQLi payload must retain a valid operator (e.g., `$gt`) after application processing; and an RCE payload must achieve partial command execution on the server. This impact-oriented approach guarantees that every validated payload is a functional exploit that authentically represents the intended attacking behavior.

Overall, both Table VII and Table VIII show that our proposed model attains a favorable balance between evasion and real exploit success. In several attack categories, our proposed model attains the highest Validate rate or achieves a strong Validate result while maintaining competitive Bypass performance. Because real-world impact requires both evasion and functionality, this joint outcome (high Bypass *and* high Validate) is the primary criterion for effectiveness.

On Table VII with ModSecurity, our model attains both high Bypass and high Validate for key attacks (e.g., SQLi: Bypass 80.16%, Validate 70.20%; SSRF: Bypass 76.06%, Validate 94.70%). For RCE and XSS the absolute Bypass percentages are lower (RCE Bypass 20.50%, XSS Bypass 7.86%), yet the Validate rates among those bypasses are extremely high (RCE Validate 88.40%, XSS Validate 92.24%). These results indicate that although fewer RCE/XSS payloads evade ModSecurity, the ones that do are highly likely to be functional exploits - an operationally important risk.

On Table VIII with SafeLine, the model often achieves the top Validate values (for example, SQLi Validate 78.91%, XSS Validate 85.26%), and it also attains very strong Bypass in some categories (reported RCE Bypass at 100% in our test run). Several baseline methods may show larger raw Bypass for particular attacks (e.g., Seq2seq shows high SQLi Bypass), but their Validate rates do not consistently match those Bypass gains. This discrepancy shows that high raw evasion alone can be misleading unless accompanied by high functional validity.

E. RQ5: How is the quality of the generated payloads quantitatively evaluated, particularly concerning redundancy and component distribution?

To answer this question, we evaluate our DEG-WAF framework against several baselines on ModSecurity WAF using the quality metrics aforementioned. Specifically, Duplicate Rate (Dupl) identifies the proportion of identical, redundant payloads through MD5 hashing, where a lower rate indicates higher novelty. Diversity (Div) measures the lexical richness using the Type-Token Ratio (the ratio of unique tokens to total tokens), with a higher value signifying more varied content. Finally, Entropy (Ent) uses Shannon Entropy to measure the unpredictability of the token distribution; higher entropy suggests a more sophisticated and less predictable payload structure. The comparative results are presented in Table IX.

As shown in Table IX, our proposed model, DEG-WAF, demonstrates consistently superior performance in generating high-quality payloads across most attack categories. For NoSQLi, XSS, and SSRF, our model achieves the best results across all three metrics: the lowest Duplicate Rate (e.g., a near-zero 0.0158 for NoSQLi), the highest Diversity (e.g., 0.984 for NoSQLi), and the highest or near-highest Entropy. This indicates an exceptional ability to produce novel, structurally varied, and unpredictable payloads for these attack types. For RCE, while Flan-t5-small shows marginally higher Entropy, our model achieves a significantly lower Duplicate Rate

TABLE VII. *Comparison of Bypass (%) and Validate (%) between our proposed model and deep-learning baselines across attack types (5,000 payloads per attack) on Modsecurity WAF*

Model	SQLi		RCE		XSS		SSRF		NoSQLi	
	Bypass (%)	Validate (%)	Bypass (%)	Validate (%)	Bypass (%)	Validate (%)	Bypass (%)	Validate (%)	Bypass (%)	Validate (%)
LSTM	65.78	47.04	20.34	65.18	3.30	89.57	64.10	77.82	47.06	53.24
Seq2seq	74.61	36.24	20.46	72.48	2.18	86.24	75.18	79.41	67.85	42.28
Flan-t5-small	63.08	60.21	19.20	42.60	6.42	72.60	70.54	74.99	71.24	27.52
SeqGAN	57.63	32.90	20.32	52.49	4.18	82.18	64.92	68.76	73.26	38.16
GPTFuzzer	75.16	64.27	20.18	81.64	2.34	78.82	74.62	83.26	69.86	58.92
DEG-WAF (Ours)	**80.16**	**70.20**	**20.50**	**88.40**	**7.86**	**92.24**	**76.06**	**94.70**	**74.70**	**77.62**

TABLE VIII. *Comparison of Bypass (%) and Validate (%) between our proposed model and deep-learning baselines across attack types (5,000 payloads per attack) on SafeLine WAF*

Model	SQLi		RCE		XSS		SSRF	
	Bypass (%)	Validate (%)	Bypass (%)	Validate (%)	Bypass (%)	Validate (%)	Bypass (%)	Validate (%)
LSTM	64.50	43.33	81.28	83.40	45.70	74.20	82.68	79.87
Seq2seq	**89.52**	54.60	91.40	**85.70**	44.74	65.62	**86.78**	82.10
Flan-t5-small	69.42	60.24	95.70	78.21	42.60	58.92	80.60	58.17
SeqGAN	63.70	48.12	86.60	72.74	52.74	69.20	82.94	68.40
GPTFuzzer	82.62	66.18	94.40	82.14	53.70	62.70	82.70	78.50
DEG-WAF (Ours)	74.17	**78.91**	**100**	81.40	**62.90**	**85.26**	84.82	**86.70**

TABLE IX. *Comparison of payload quality metrics (Duplicate Rate, Diversity, Entropy) between our proposed model and baselines on ModSecurity WAF*

Model	SQLi			RCE			XSS			SSRF			NoSQLi		
	Dupl	Div	Ent	Dupl	Div	Ent	Dupl	Div	Ent	Dupl	Div	Ent	Dupl	Div	Ent
LSTM	0.753	0.247	8.214	0.551	0.449	9.102	0.603	0.397	9.055	0.582	0.418	9.211	0.625	0.375	8.832
Seq2seq	0.721	0.279	8.533	0.512	0.488	9.541	0.556	0.444	9.387	0.549	0.451	9.601	0.598	0.402	9.017
Flan-t5-small	0.586	0.414	10.112	0.157	0.843	12.155	0.114	0.886	11.932	0.141	0.859	11.841	0.093	0.907	12.088
SeqGAN	0.654	0.346	9.802	0.301	0.699	10.881	0.259	0.741	11.034	0.283	0.717	10.952	0.226	0.774	11.213
GPTFuzzer	0.495	0.505	10.351	0.124	0.876	11.988	0.071	0.929	12.146	0.105	0.895	12.033	0.052	0.948	12.159
DEG-WAF (Ours)	**0.512**	**0.488**	**10.534**	**0.0898**	**0.910**	**12.101**	**0.042**	**0.958**	**12.203**	**0.083**	**0.917**	**12.101**	**0.0158**	**0.984**	**12.255**

(0.0898 vs. 0.157) and the highest Diversity (0.910), making its payloads far less repetitive.

In the case of SQLi, the results highlight a nuanced trade-off. While GPTFuzzer achieves a slightly lower Duplicate Rate, our model produces payloads with the highest Entropy (10.534). This suggests that our payloads, though occasionally sharing similarities, possess the most balanced and unpredictable distribution of keywords, a critical feature for evading advanced, statistics-based detection systems. Overall, the data confirms that our model excels at creating a diverse and sophisticated arsenal of attacks that avoid the common pitfall of repetitive, easily-fingerprinted patterns seen in other models.

F. Case Study: Successful Evasive Payload and Analysis

To further demonstrate the effectiveness of the proposed framework, we present a representative example of an evasive payload that

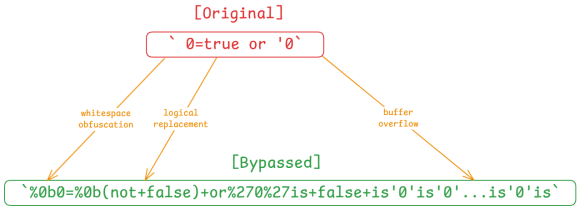


Figure 2. Tranformation of the SQL Injection payload

successfully bypassed the ModSecurity WAF configured with the OWASP Core Rule Set (CRS), as illustrated in Fig. 2. The generated payload targets a Boolean-based SQL injection vulnerability and is constructed as follows:

```
%0b0=%0b(not+false)+or%270%27is+false
      +is'0'is'0'...is'0'is
```

As depicted in Fig. 2, the original (non-evasive) form of the payload was simply `0=true or '0'`, which is a canonical injection pattern. However, such a straightforward

formulation would likely be detected and blocked by the WAF due to its direct use of logical operators (e.g., `or`), explicit Boolean comparisons (e.g., `0=true`), and the presence of an unescaped or unclosed single quote character. In contrast, the transformed version introduces obfuscation via encoding and structural perturbation while preserving the original attack semantics, thereby successfully evading detection.

The transformation of the original payload in our framework is achieved through the integration of three complementary obfuscation techniques, each targeting a specific weakness in common WAF detection strategies.

First, whitespace obfuscation is applied by substituting standard delimiters (e.g., spaces or tabs) with less frequently used but syntactically valid characters, such as the vertical tab encoded as `%0b`. This form of obfuscation exploits the fact that many WAFs restrict their inspection to a predefined set of whitespace characters, overlooking rare variants like `%0b`. As a result, the payload bypasses pattern-based filtering rules without altering its functional semantics.

Second, the method employs logical replacement, wherein common Boolean expressions are restructured into semantically equivalent but syntactically divergent forms. For example, the expression `true` is substituted with `not false`, reducing the likelihood of triggering signature-based detection that relies on matching well-known tokens or operators. This transformation maintains the payload's logical intent while increasing its evasiveness through linguistic variation.

Finally, a form of length-based evasion is utilized by artificially inflating the payload size - often exceeding 1000 characters. This tactic exploits practical constraints in WAF inspection mechanisms, where resource-saving heuristics may limit the depth or extent of analysis for overly long inputs. By exceeding such thresholds, the payload may avoid full inspection, thereby increasing its chance of successful delivery to the backend application.

Together, these transformations demonstrate how structural manipulation - grounded in syntactic permissiveness and behavioral inference - can effectively bypass modern WAF defenses while preserving attack semantics.

VII. Discussion

While our proposed DEG-WAF framework demonstrates strong performance in bypassing both rule-based and machine learning-based WAFs, it still has certain limitations. First, the current model is limited to the OPT-125M backbone, which may restrict the semantic capacity and depth of payload generation. Second, the reward model is trained using binary WAF verdicts (bypass/block), which does not capture nuanced behaviors such as partial blocking, latency responses, or error codes. Additionally, the evaluation scope is limited to two open-source WAFs (ModSecurity and SafeLine), and may not fully represent the complexity of commercial-grade WAFs. The framework also assumes a black-box setting without handling authenticated or session-aware WAF behaviors.

To further improve the system, future work can explore scaling the LLM using larger architectures like GPT-J, LLaMA, or Mistral to increase generation fidelity. RL can also be enhanced by adopting advanced variants such as PPO-Lagrangian or Self-Imitation Learning. Moreover, integrating post-WAF exploit agents would help verify whether successful bypasses lead to real-world impact. Expanding evaluations to commercial WAFs (e.g., Cloudflare WAF, AWS WAF) and incorporating multi-metric benchmarks—including exploitability and semantic correctness—can provide a more comprehensive understanding. Automating the full pipeline from generation to validation will be critical for real-world deployment.

VIII. Conclusion

This paper presents DEG-WAF, an AI-driven framework that leverages large language models (LLMs) and RL to generate evasive web attack payloads capable of bypassing modern WAFs. By combining the OPT-125M model with PPO and A2C algorithms, as well as integrating a reward agent and grammar-guided sampler, our system significantly improves bypass rates across diverse attack types including SQLi, RCE, SSRF, XSS, and NoSQLi. Experimental results on ModSecurity and SafeLine demonstrate the effectiveness and adaptability of our approach. The framework outperforms the baseline model, maintains structural validity, and generalizes well

under black-box conditions. Overall, this work offers a practical foundation for building automated security testing systems that assess WAF robustness in adversarial environments.

Acknowledgement

This research is funded by Vietnam National University Ho Chi Minh City (VNU-HCM) under grant number NCM2025-26-01.

References

[1] O. Fredj, O. Cheikhrouhou, M. Krichen, H. Hamam, and A. Derhab, "An owasp top ten driven survey on web application protection methods," 11 2020.

[2] V. Clincy and H. Shahriar, "Web application firewall: Network security models and configuration," in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 01, 2018, pp. 835–836. DOI: 10.1109/COMPSAC.2018.00144

[3] A. Coscia, V. Dentamaro, S. Galantucci, A. Maci, and G. Pirlo, "Progesi: a proxy grammar to enhance web application firewall for sql injection prevention," *IEEE Access*, vol. 12, pp. 107 689–107 703, 08 2024. DOI: 10.1109/ACCESS.2024.3438092

[4] N. N. Thanh, V.-G. Ung, P. T. Duy, and V.-H. Pham, "A study on adversarial attacks for benchmarking deep learning-based web application firewalls," in *2024 RIVF International Conference on Computing and Communication Technologies (RIVF)*. IEEE, 2024, pp. 151–155.

[5] A. Valenza, L. Demetrio, G. Costa, and G. Lagorio, "Waf-a-mole: An adversarial tool for assessing ml-based wafs," *SoftwareX*, vol. 11, p. 100367, 2020. DOI: https://doi.org/10.1016/j.softx.2019.100367

[6] H. Liang, X. Li, D. Xiao, J. Liu, Y. Zhou, A. Wang, and J. Li, "Generative pre-trained transformer-based reinforcement learning for testing web application firewalls," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 1, pp. 309–324, 2024. DOI: 10.1109/TDSC.2023.3252523

[7] D. Leung, O. Tsai, K. Hashemi, B. Tayebi, and M. A. Tayebi, "Xploitsql: Advancing adversarial sql injection attack generation with language models and reinforcement learning," in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, ser. CIKM '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 4653–4660. DOI: 10.1145/3627673.3680102

[8] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain, and J. Gao, "Large language models: A survey," . , *https://arxiv.org/abs/2402.06196*

[9] H. Xu, S. Wang, N. Li, K. Wang, Y. Zhao, K. Chen, T. Yu, Y. Liu, and H. Wang, "Large language models for cyber security: A systematic literature review," *arXiv preprint arXiv:2405.04760*, 2024.

[10] V. Babaey and A. Ravindran, "Gensqli: A generative artificial intelligence framework for automatically securing web application firewalls against structured query language injection attacks," *Future Internet*, vol. 17, no. 1, 2025. DOI: 10.3390/fi17010008

[11] D. Miczek, D. Gabbireddy, and S. Saha, "Leveraging llm to strengthen ml-based cross-site scripting detection," . , *https://arxiv.org/abs/2504.21045*

[12] Z. Gui, E. Wang, B. Deng, M. Zhang, Y. Chen, S. Wei, W. Xie, and B. Wang, "Sqligpt: Evaluating and utilizing large language models for automated sql injection black-box detection," *Applied Sciences*, vol. 14, no. 16, 2024. DOI: 10.3390/app14166929

[13] V. Babaey and A. Ravindran, "Genxss: an ai-driven framework for automated detection of xss attacks in wafs," . , *https://arxiv.org/abs/2504.08176*

[14] H. Kheddar, D. W. Dawoud, A. I. Awad, Y. Himeur, and M. K. Khan, "Reinforcement-learning-based intrusion detection in communication networks: A review," *IEEE Communications Surveys & Tutorials*, 2024.

[15] M. Ghasemi and D. Ebrahimi, "Introduction to reinforcement learning," . , *https://arxiv.org/abs/2408.07712*

[16] S. Finistrella, S. Mariani, and F. Zambonelli, "Multi-agent reinforcement learning for cybersecurity: Classification and survey," *Intelligent Systems with Applications*, p. 200495, 2025.

[17] C. Folini and I. Ristic, *ModSecurity Handbook, Second Edition*, 2nd ed. London, GBR: Feisty Duck, 2017.

[18] Chaitin Technology, "Safeline web application firewall documentation," 2023. , Access date: 10/6/2025, *https://docs.waf.chaitin.com*

[19] Wargio, "Naxsi: Nginx anti xss and sql injection," 2023. , Access date: 10/6/2025, *https://github.com/wargio/naxsi*

[20] OWASP ModSecurity Core Rule Set Team, "Owasp core rule set (crs)," 2023. , Access date: 9/6/2025, *https://github.com/coreruleset/coreruleset*

[21] S. Dhote, A. Magdum, S. Singh, and D. Raigar, "Ml based web application firewall for signature and anomaly detection using feature extraction," in *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2024, pp. 1–6. DOI: 10.1109/ICCCNT61001.2024.10725511

[22] K. S. Kalyan, "A survey of gpt-3 family large language models including chatgpt and gpt-4," *Natural Language Processing Journal*, vol. 6, p. 100048, 2024. DOI: https://doi.org/10.1016/j.nlp.2023.100048

[23] G. Yenduri, R. M, C. S. G, S. Y, G. Srivastava, P. K. R. Maddikunta, D. R. G, R. H. Jhaveri, P. B, W. Wang, A. V. Vasilakos, and T. R. Gadekallu, "Generative pre-trained transformer: A comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions," . , *https://arxiv.org/abs/2305.10435*

[24] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, and L. Zettlemoyer, "Opt: Open pre-trained transformer language models," . , *https://arxiv.org/abs/2205.01068*

[25] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," . , *https://arxiv.org/abs/2302.13971*

[26] H. Zhou, C. Hu, Y. Yuan, Y. Cui, Y. Jin, C. Chen, H. Wu, D. Yuan, L. Jiang, D. Wu, X. Liu, J. Zhang, X. Wang, and J. Liu, "Large language model (llm) for telecommunications: A comprehensive survey on principles, key techniques, and opportunities," *IEEE Communications Surveys Tutorials*, vol. 27, no. 3, pp. 1955–2005, 2025. DOI: 10.1109/COMST.2024.3465447

[27] Y. Yao, J. Duan, K. Xu, Y. Cai, Z. Sun, and Y. Zhang, "A survey on large language model (llm) security and privacy: The good, the bad, and the ugly," *High-Confidence Computing*, vol. 4, no. 2, p. 100211, Jun. 2024. DOI: 10.1016/j.hcc.2024.100211

[28] J. Zhang, H. Bu, H. Wen, Y. Liu, H. Fei, R. Xi, L. Li, Y. Yang, H. Zhu, and D. Meng, "When llms meet cybersecurity: A systematic literature review," *Cybersecurity*, vol. 8, no. 1, p. 55, 2025.

[29] A. Ramé, N. Vieillard, L. Hussenot, R. Dadashi, G. Cideron, O. Bachem, and J. Ferret, "Warm: On the benefits of weight averaged reward models," . , *https://arxiv.org/abs/2401.12187*

[30] V. Atlidakis, R. Geambasu, P. Godefroid, M. Polishchuk, and B. Ray, "Pythia: Grammar-based fuzzing of rest apis with coverage-guided feedback and learning-based mutations," *arXiv preprint arXiv:2005.11498*, 2020.

[31] P. Godefroid, H. Peleg, and R. Singh, "Learn&fuzz: Machine learning for input fuzzing," in *2017 32nd IEEE/ACM International Conference on Automated*

Software Engineering (ASE). IEEE, 2017, pp. 50–59.

[32] P. Srivastava and M. Payer, "Gramatron: effective grammar-aware fuzzing," in *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 244–256. DOI: 10.1145/3460319.3464814

[33] Z. Qu, X. Ling, T. Wang, X. Chen, S. Ji, and C. Wu, "Advsqli: Generating adversarial sql injections against real-world waf-as-a-service," *IEEE Transactions on Information Forensics and Security*, vol. 19, p. 2623–2638, 2024. DOI: 10.1109/tifs.2024.3350911

[34] K. Li, H. Yang, and W. Visser, "Evolutionary multi-task injection testing on web application firewalls," . , *https://arxiv.org/abs/2206.05743*

[35] C. Wu, J. Chen, S. Zhu, W. Feng, R. Du, and Y. Xiang, "Wafbooster: Automatic boosting of waf security against mutated malicious payloads," . , *https://arxiv.org/abs/2501.14008*

[36] F. Yang, W. Zhou, Z. Liu, D. Zhao, and D. Held, "Reinforcement learning in a safety-embedded mdp with trajectory optimization," . , *https://arxiv.org/abs/2310.06903*

[37] M.-A. Chadi and H. Mousannif, "Understanding reinforcement learning algorithms: The progress from basic q-learning to proximal policy optimization," . , *https://arxiv.org/abs/2304.00026*

[38] OpenAI, "Spinning up - proximal policy optimization (ppo)," 2024. , Access date: 10/6/2025, *https://spinningup.openai.com/en/latest/algorithms/ppo.html*

[39] GeeksforGeeks Contributors, "Actor-critic algorithm in reinforcement learning," 2023. , Access date: 10/6/2025, *https://www.geeksforgeeks.org/machine-learning/actor-critic-algorithm-in-reinforcement-learning/*

[40] Swisskyrepo, "Payloads all the things," 2023. , Access date: 10/6/2025, *https://github.com/swisskyrepo/PayloadsAllTheThings*

About the authors



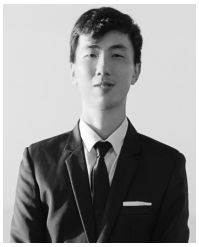
Tran Gia Bao
Workplace:
University of Information Technology, Vietnam National University Ho Chi Minh City (VNU-HCM)
Email: 22520119@gm.uit.edu.vn
Education: Final-year student majoring in Information Security
Recent research interests: Web security, penetration testing.

Tên tác giả: **Trần Gia Bảo**
Cơ quan công tác: Trường Đại học Công nghệ Thông tin, Đại học Quốc gia thành phố Hồ chí Minh, Việt Nam
Email: 22520119@gm.uit.edu.vn
Quá trình đào tạo: Sinh viên năm cuối chuyên ngành An toàn Thông tin (Chương trình Cử nhân Tài năng)
Hướng nghiên cứu hiện nay: An toàn ứng dụng web, kiểm thử xâm nhập.



Dinh Cong Duc
Workplace:
University of Information Technology, Vietnam National University Ho Chi Minh City (VNU-HCM)
Email: 22520262@gm.uit.edu.vn
Education: Final-year student majoring in Information Security
Recent research interests: Web security, penetration testing.

Tên tác giả: **Đinh Công Đức**
Cơ quan công tác: Trường Đại học Công nghệ Thông tin, Đại học Quốc gia thành phố Hồ chí Minh, Việt Nam
Email: 22520262@gm.uit.edu.vn
Quá trình đào tạo: Sinh viên năm cuối chuyên ngành An toàn Thông tin (Chương trình Cử nhân Tài năng)
Hướng nghiên cứu hiện nay: An toàn ứng dụng web, kiểm thử xâm nhập.



Phan The Duy
Workplace:
University of Information Technology, Vietnam National University Ho Chi Minh City (VNU-HCM)
Email: duypt@uit.edu.vn
Education: PhD in Information Technology, specialized in Information Security
Recent research interests: Penetration Testing, Web security, Smart contract security, malware analysis.

Tên tác giả: **Phan Thế Duy**
Cơ quan công tác: Trường Đại học Công nghệ thông tin, Đại học Quốc gia Thành phố Hồ chí Minh.
Email: duypt@uit.edu.vn
Quá trình đào tạo: Tiến sĩ ngành Công nghệ thông tin, chuyên ngành An toàn thông tin
Hướng nghiên cứu hiện nay: Kiểm thử thâm nhập, an toàn ứng dụng web, bảo mật Hợp đồng thông minh, phân tích mã độc.