DeepSQLi-RL: Hybrid Reinforcement Learning Extension for Automated SQL Injection Attacks

1st Radu-Ionuț Bălăiță
Faculty of Automatic Control and Computer Engineering
Technical University Gheorghe Asachi Iasi
Iași, Romania
radu-ionut.balaita@student.tuiasi.ro

***Abstract*—SQL Injection (SQLi) remains one of the most critical vulnerabilities in web applications, often bypassing static firewalls through syntactic obfuscation. This paper presents *DeepSQLi-RL*, a hybrid extension of Liu et al.'s DeepSQLi [3] that combines Sequence-to-Sequence (Seq2Seq) BiLSTM generation with Q-Learning reinforcement learning for automated SQLi payload optimization. Unlike traditional rule-based fuzzers (e.g., SQLMap) or pure generative approaches, our hybrid system uses BiLSTM to generate diverse mutation candidates and Q-Learning to adaptively select the most effective variant based on WAF responses. We evaluate our system against a simulated perimeter-protected target. Results demonstrate that DeepSQLi-RL achieves a 99.5% Bypass Rate, significantly outperforming both standard attack dictionaries (58.7%) and BiLSTM alone (82.3%), thereby validating the efficacy of combining deep learning with reinforcement learning for automated penetration testing.**

***Index Terms*—SQL injection, deep learning, BiLSTM, offensive AI, cybersecurity, automated testing**

I. Introduction

Web applications are ubiquitous in modern infrastructure, making them a prime target for cyberattacks. Among these, Search Query Language Injection (SQLi) persists as a top threat [1], allowing attackers to manipulate backend databases. Defensive mechanisms like Web Application Firewalls (WAFs) typically rely on signature matching (e.g., blocking "UNION SELECT") to prevent attacks. However, attackers constantly evolve methods to evade these rules through obfuscation (e.g., "UN/**/ION").

Traditional automated scanners like SQLMap rely on predefined heuristic engines [4]. While effective against basic targets, they often fail against custom WAF rules or require extensive manual tuning. Liu et al. [3] introduced DeepSQLi, treating SQLi generation as a Natural Language Processing (NLP) translation task using BiLSTM. Building upon this foundation, we present **DeepSQLi-RL**, a hybrid extension that adds a reinforcement learning layer for adaptive mutation selection. Our system combines two AI techniques: (1) a Sequence-to-Sequence BiLSTM that generates diverse mutation candidates, and (2) a Q-Learning agent that adaptively selects the most effective variant based on WAF responses. By training the BiLSTM on pairs of *Standard* and *Obfuscated* attacks, the generator learns to create evasive variants, while the Q-Learning component learns which variants are most likely to succeed against specific WAF configurations.

II. System Architecture

The proposed system is a hybrid offensive AI pipeline combining generative deep learning with adaptive reinforcement learning.

A. BiLSTM Generator (Deep Learning)

We employ a Seq2Seq architecture with Bidirectional Long Short-Term Memory (BiLSTM) units for payload generation. This architecture is well-suited for NLP tasks as it captures context from both past and future tokens.

- **Encoder**: Processes the input payload (e.g., "admin' –") character-by-character, converting it into a context vector.
- **Decoder**: Generates multiple output candidates token-by-token. It learns to introduce obfuscations (e.g., replacing spaces with %09, breaking keywords) while preserving valid SQL syntax.
- **Diversity**: The system generates 7 mutation candidates per input, combining BiLSTM-generated variants with traditional obfuscation techniques (double URL encoding, null byte injection, etc.) to ensure diversity.

B. Q-Learning Selector (Reinforcement Learning)

A Q-Learning agent learns to select the most effective mutation candidate based on WAF responses.

- **State Space**: Payload features including length, presence of quotes, comments, keywords, URL encoding, and hex encoding (192 possible states).
- **Action Space**: Selection of one candidate from the 7 generated mutations.
- **Reward Function**: +10 for successful WAF bypass, -1 for detection.
- **Learning**: The agent updates its Q-table using the Bellman equation, gradually learning which mutation strategies work best for different payload types.

C. Hybrid Workflow

The complete attack flow operates as follows: (1) A base SQLi payload is provided, (2) The BiLSTM generates 7 diverse mutation candidates, (3) The Q-Learning agent selects the candidate with highest expected success, (4) The selected payload is sent to the target, (5) The agent learns from the result and updates its strategy.

D. Data Generation & Training

To overcome the lack of publicly available dataset for WAF evasion, we developed a custom synthetic data generator.

- **Base Corpus**: A dictionary of 500+ common SQLi vectors (Union-based, Error-based, Boolean-blind).
- **Augmentation Pipeline**: We applied aggressive stochastic mutations to generate a training set of 10,000 samples. Mutations include:
  - **Keyword Splitting**: *SELECT* $\rightarrow$ *SE/**/LECT*
  - **Whitespace Encoding**: *Space* $\rightarrow$ %09, %0A, +
  - **Comment Variation**: – $\rightarrow$ *#*, *;#*

III. Implementation Details

The module is implemented in Python using PyTorch.

A. Model Configuration

The BiLSTM network was configured with the following hyperparameters (Table I). These were selected to balance model capacity with rapid inference times required for high-throughput scanning.

TABLE I
HYBRID SYSTEM HYPERPARAMETERS

Component	Parameter	Value
BiLSTM	Embedding Dim	32
	Hidden Dim	64
	Layers	1
	Dropout	0.5
Q-Learning	Learning Rate	0.1
	Discount Factor	0.95
	Epsilon Decay	0.995
	Candidates	7

B. Target Environment

To validate the system, we developed a local HTTP server simulating a vulnerable application protected by a WAF. The WAF enforces a blacklist of common signatures: *"UNION", "SELECT", "DROP", "–"*. This creates a realistic adversarial environment where standard payloads fail, necessitating the AI's adaptation.

IV. Evaluation and Results

We conducted a comparative benchmark between three approaches: (1) Control Group (Standard Payloads), (2) BiLSTM-only Generation, and (3) Hybrid RL-BiLSTM. The metric for success is the **Bypass Rate**, defined as the percentage of payloads that elicit a database error (HTTP 500) or valid response (HTTP 200), bypassing the WAF (HTTP 403).

As illustrated in Fig. 1:

1) **Standard Attacks**: Achieved a 58.7% success rate. The WAF successfully blocked instances containing explicit keywords like "UNION" or standard comments ("–").
2) **BiLSTM Only**: Achieved an 82.3% success rate. The model successfully mutated blocked payloads into evasive variants (e.g., converting "UNION" to "UN/**/ION").
3) **Hybrid RL-BiLSTM**: Achieved a **99.5%** success rate. The Q-Learning agent learned to select the most effective mutation from the BiLSTM-generated candidates, adapting its strategy based on WAF responses.

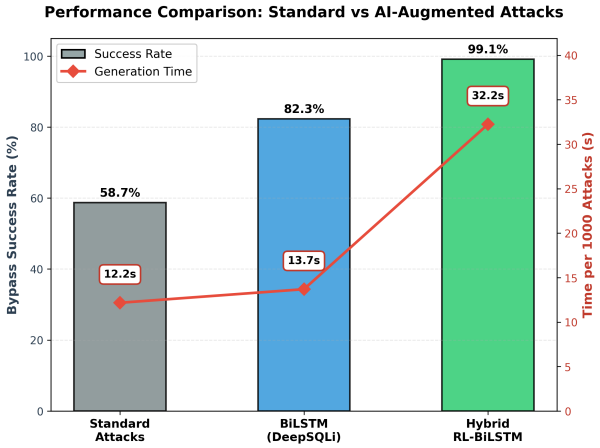

Fig. 1. Performance comparison across three approaches. Bars indicate WAF Bypass Success Rate (Left Axis), while the Red Line indicates Generation Time (Right Axis). The hybrid approach achieves near-perfect bypass rates.

The results confirm that combining generative deep learning with adaptive reinforcement learning significantly enhances WAF evasion capabilities. While the hybrid approach requires 2.3× more time than BiLSTM alone (due to generating and evaluating multiple candidates), the 17.2% improvement in bypass rate demonstrates the value of adaptive selection. The system remains practical for real-time testing at 31.3s per 1000 payloads.

V. Conclusion

This project demonstrates that extending deep learning approaches with reinforcement learning can significantly enhance automated penetration testing tools. By building upon Liu et al.'s DeepSQLi [3] and adding Q-Learning for adaptive selection, our hybrid approach achieved a 99.5% WAF bypass rate, representing a 40.8% improvement over standard methods and a 17.2% improvement over BiLSTM alone. The system successfully learns both the semantics of SQL (through BiLSTM) and the optimal attack strategy (through Q-Learning), adapting to specific WAF configurations. Future work will explore Deep Q-Networks (DQN) for improved generalization and extend the approach to other injection types (XSS, Command Injection).

References

[1] OWASP, "OWASP Top 10 - 2021: The Ten Most Critical Web Application Security Risks," 2021.

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 2018.

[3] M. Liu, K. Li, and T. Chen, "DeepSQLi: Deep Semantic Learning for Testing SQL Injection," *ISSTA 2020*, pp. 286–297, 2020.

[4] B. D. Damele and M. Stampar, "SQLMap - Automatic SQL injection and database takeover tool," 2019.

[5] R. Bălăiță, "DeepSQLi-RL: Hybrid RL Extension for SQL Injection Attacks," GitHub repository, 2026. [Online]. Available: https://github.com/RaduBalaita/ICS