RL-Semantic: RL-Enhanced Semantic Fuzzing for Bluetooth Low Energy Protocols

1st Qinghe Li
Information Engineering University
Zhengzhou, China
suami0027@outlook.com

2nd Ruijie Cai
Information Engineering University
Zhengzhou, China
wsxcrj@163.com

3rd Xiaokang Yin
Information Engineering University
Zhengzhou, China
yxksjtu@sjtu.edu.cn

4th Yaobin Xie
Information Engineering University
Zhengzhou, China
yb_xie@163.com

5th Jiahan Liu
Information Engineering University
Zhengzhou, China
liujiahan0628@outlook.com

6th Shengli Liu
Information Engineering University
Zhengzhou, China
mr_shengliliu@163.com

7th Lukai Li
Information Engineering University
Zhengzhou, China
1586727291@qq.com

***Abstract*—Existing protocol fuzzing tools struggle to achieve comprehensive Bluetooth Low Energy (BLE) coverage due to limited understanding of its complex state machines and inefficient test strategy selection. We propose RL-Semantic, a framework integrating protocol-aware semantic mutation strategies with Reinforcement Learning (RL) for adaptive strategy selection. We formulate the BLE fuzzing process as a Markov Decision Process (MDP) and model the protocol stack as a four-layer hierarchical state machine. Our approach then employs five semantic mutation strategies with DQN-based adaptive selection, using two-phase training that combines offline supervised learning and online RL. Evaluation on ten simulated BLE targets over 360-minute runs, RL-Semantic is compared against Field, AFL, and Random baselines. RL-Semantic achieves an average of 266 state transitions, substantially outperforming Field with 55 transitions, AFL with 61, and Random with 63. Notably, RL-Semantic demonstrates decisive advantages at security-critical layers, achieving a 6.9 times improvement at GATT layer and transitioning from zero to non-zero coverage at SMP layer where Field fails to trigger any transitions. Ablation experiments validate that semantic strategies constitute the primary performance driver while RL provides incremental optimization through adaptive strategy selection.**

***Index Terms*—reinforcement learning, deep Q network, protocol fuzzing, semantic mutation, bluetooth low energy.**

I. Introduction

BLE has emerged as the dominant protocol for low-power IoT, medical, and wearable devices, creating a significant attack surface as billions of devices become online. The BLE protocol stack involves complex interactions across four layers: the Link Layer (LL), Generic Attribute Profile (GATT), Logical Link Control and Adaptation Protocol (L2CAP), and Security Manager Protocol (SMP). This stateful complexity has led to numerous high-severity vulnerabilities, including buffer overflows, integer underflows, and race conditions, highlighting the need for effective state-aware security testing.

Recent testing efforts for BLE have explored various technical directions. Combinatorial approaches [1] attempt to integrate symbolic execution with fuzzing to enhance coverage but are often constrained by the complex state space of BLE. Protocol state-aware fuzzing [2] has begun incorporating state machine modeling, yet its ability to fully explore state remains limited. Although systematic threat analyses [3] have identified vulnerabilities in BLE stacks, they generally lack automated mechanisms for vulnerability discovery.

In this context, BLuEMan [4] represents a significant breakthrough by introducing a stateful, field-based mutation method, achieving throughput improvements by a factor of 18 to 162 across different platforms. BLuEMan establishes the first practical framework that combines packet sequence tracking with protocol semantics for field mutation, demonstrating that stateful fuzzing can systematically navigate complex protocol state spaces. It has uncovered four novel vulnerabilities, CVE-2023-4424, CVE-2024-3077, CVE-2024-3332, and CVE-2024-4785, all rooted in protocol state machine flaws, validating the effectiveness of state-aware testing and establishing a strong baseline for protocol fuzzing research.

Despite these advances, a fundamental limitation common to existing protocol fuzzing approaches is the use of static, context-agnostic mutation strategies, which suffer from a semantic gap. By applying mutations independent of protocol state, such approaches generate packets that frequently violate basic specifications, leading to rejection by the target implementation before reaching deeper logic. This semantic gap arises because packet-only fuzzing cannot access the state-dependent validity constraints defined in the Bluetooth Core Specification [5]. Recent surveys on protocol fuzzing [6] highlight this as a key challenge: state-agnostic mutations waste fuzzing cycles on invalid packets that never exercise

meaningful state transitions or boundary conditions.

While state-of-the-art methods such as BLuEMan [4] introduce manual semantic mutation strategies, they remain constrained by a static policy selection mechanism. BLuEMan's field-based mutations achieve substantial throughput gains, yet the selection of which mutation strategy to apply is predetermined rather than adaptive, failing to account for the varying effectiveness of different strategies across protocol states or to leverage feedback from previous test cases. This static approach limits the ability to focus mutations on strategies most effective for specific states, such as boundary testing for parameter validation or transition testing for state machine logic. To address this gap, we introduce RL-Semantic, which combines formal protocol modeling with RL to dynamically select mutation strategies based on protocol state and real-time coverage feedback. Extensive evaluation across ten BLE targets demonstrates RL-Semantic's effectiveness, achieving 266 average state transitions compared to 55 for BLuEMan, representing a 4.8 times improvement.

In this work, we make the following contributions:

- **W**e model BLE as a four-layer, 44-state hierarchical state machine and formulate protocol fuzzing as a sequential decision-making problem optimized via deep RL, establishing a rigorous theoretical foundation for state-aware protocol fuzzing that enables systematic exploration of protocol state spaces.
- **T**hrough systematic analysis of the Bluetooth Core Specification, we design five semantic mutation strategies targeting conformance, boundary, transition, and robustness testing, enabling decisive advantages at security-critical protocol layers with a 6.9 times improvement at GATT and non-zero coverage at SMP where traditional methods fail.
- **W**e employ a two-phase trained DQN policy selector that dynamically adapts mutation strategy selection based on protocol states and coverage feedback, with ablation studies demonstrating that RL optimizes strategy selection in specific states and enhances performance beyond static policy approaches.

II. Related Work

Protocol fuzzing has evolved significantly from early coverage-guided approaches to sophisticated semantic-aware methods leveraging protocol specifications. Recent advances in machine learning have further enabled adaptive fuzzing through RL. We structure our review of related work along three technical dimensions that represent this evolution: protocol fuzzing techniques, semantic-aware fuzzing approaches, and RL applications to fuzzing.

A. Protocol fuzzing

Coverage-guided fuzzing has been extensively applied to protocol testing to discover vulnerabilities in protocol implementations. This approach typically focuses on code coverage metrics, which guide the formulation of test cases by tracking the boundaries of the executed code.

AFL [7] pioneered edge-coverage guidance, demonstrating notable effectiveness in vulnerability discovery. AFL++ [8] advanced AFL through enhanced mutation techniques and optimization strategies. AFLNet [9] extended AFL to protocol fuzzing by modeling protocol messages as sequences and incorporating stateful coverage. Recent surveys [6], [10] provide comprehensive overviews of coverage-guided grey-box fuzzing. Structure-aware fuzzing [11] enhances protocol testing by modeling input structure, yet lacks deep protocol semantic understanding. Directed grey-box fuzzing [12] effectively reaches specific target locations, yet its application to protocol state exploration remains limited.

State-aware approaches have emerged to address protocol state exploration. BLuEMan [4] introduced state-tracking for BLE protocol fuzzing, while McFluffin [13] explores Bluetooth controller fuzzing through state machine inference. Logos [14], CSFuzzer [15], and StatePre [16] enhance state-exploration efficiency through execution logs, context-aware mutation, and large language models respectively.

However, existing approaches face three critical limitations. First, they primarily focus on code coverage [17], which tracks executed code edges but inadequately captures protocol state transitions critical for uncovering state-machine vulnerabilities. Second, BLuEMan relies on heuristic mutation strategies without leveraging protocol specifications or semantic constraints, constraining its capability to systematically explore protocol state transitions. Third, existing methods either depend on fixed heuristic rules or require resource-intensive pre-trained models.

B. Semantic-Aware Fuzzing

Semantic-aware fuzzing approaches leverage protocol specifications and domain knowledge to enhance test case generation efficiency. These methods aim to generate structurally valid and semantically meaningful test inputs that can effectively exercise protocol implementations.

Constraint-based fuzzing [18] utilizes protocol constraints to guide test-case generation. Symbolic execution [19] techniques generate precise test inputs through constraint solving. Several works have addressed stateful protocol fuzzing specifically. SYZTRUST [20] introduces a state-aware fuzzing method for IoT trusted operating systems. IoTFuzzSentry [21] employs protocol-guided mutation to achieve systematic state space exploration for IoT devices. LLM-assisted fuzzing [22] leverages large language models for IoT protocol testing.

Despite these advancements, existing semantic-based policy-selection methods predominantly rely on fixed heuristic rules. This lack of adaptability constrains efficiency, as the optimal policy varies with protocol state, target objectives, and fuzzing phase. Recent research on Matter Protocol fuzz [23] underscores the importance of adaptive testing, yet requires manual configuration.

In contrast, our work employs RL for adaptive semantic policy selection, whereas existing semantic-aware approaches rely on static heuristic rules or manual configuration.

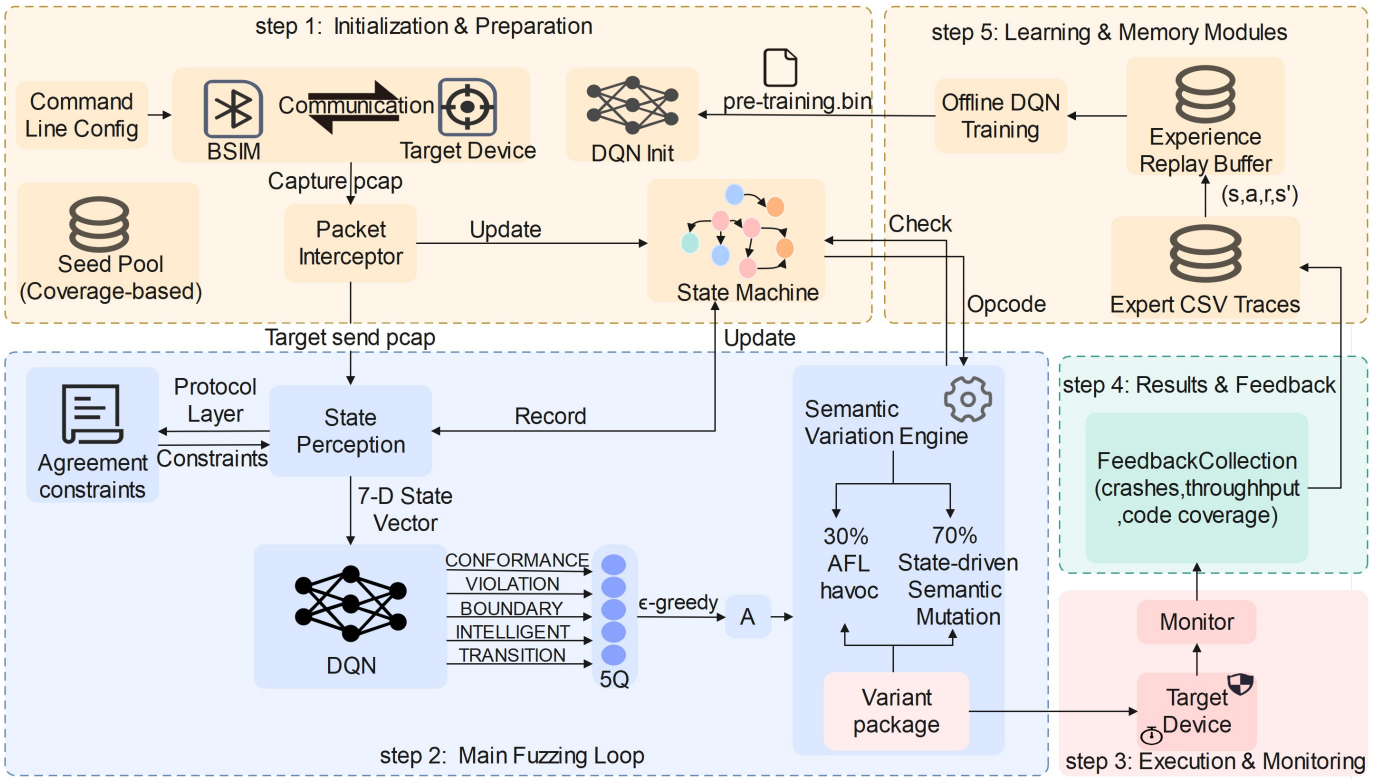

Fig. 1. Work Flow: After initialization, the main loop encodes the protocol state into a 7-D vector; the DQN selects a mutation action via ϵ-greedy, which the semantic variation engine executes and sends to the target. Execution feedback is stored for offline DQN training to continuously refine policy.

C. Reinforcement Learning for Fuzzing

RL has emerged as a promising technique for guiding fuzzing processes by learning optimal mutation strategies through environmental feedback. A recent survey [10] synthesizes the integration of deep neural models with coverage-guided fuzzing.

RLF [24] targets smart-contract fuzzing and achieves significant efficiency gains through RL-guided test case generation. APIRL [25] employs deep RL for REST API fuzzing, showing that RL can effectively navigate complex state spaces in API testing. Other approaches address web browsers [26] and deep neural networks [27], with recent surveys [28], [29] demonstrating RL's effectiveness in neural network testing.

However, existing RL-based fuzzing approaches face limitations when applied to protocol state exploration. While RL-GFuzz [30] applies DQN to protocol fuzzing, it operates at the binary level without leveraging high-level protocol semantics for adaptive strategy selection. The fundamental differences between protocol fuzzing and other domains manifest in three aspects. Protocol fuzzing requires maintaining hierarchical state machine models across multiple protocol layers, whereas binary fuzzing typically tracks simpler state representations. The action space in protocol fuzzing should comprise semantic mutation strategies rather than low-level bit manipulation operations, enhancing interpretability and learning efficiency. Reward design must account for protocol-specific objectives such as state transition validity and coverage efficiency, rather than code coverage alone. To the best of our knowledge, our work represents the first systematic application of RL to semantic-aware protocol-state exploration.

III. Approach

In this section, we introduce the RL-Semantic framework and design components. Fig. 1 illustrates the complete workflow, which integrates deep RL with protocol state modeling for intelligent BLE protocol fuzzing. The framework operates through five coordinated phases across three concurrent processes.

During initialization, the system reads configuration from command-line arguments and launches three concurrent processes: Bluetooth Simulator (BSIM), BLE Target and Interceptor, a packet sniffering. Simultaneously, it initializes two core components: the BLE state machine tracking protocol states across four layers, and the DQN network with either pre-trained or randomly initialized weights.

During corpus collection, BSim simulates normal BLE device communication while Interceptor captures packets. Each captured packet updates the state machine, recording protocol progression. These packets constitute the initial corpus for subsequent mutation.

The fuzzing loop constitutes the core iterative process executing five steps per cycle. First, Interceptor retrieves packets from the message queue and updates the BLE state machine by parsing protocol layers. Second, the current state is encoded

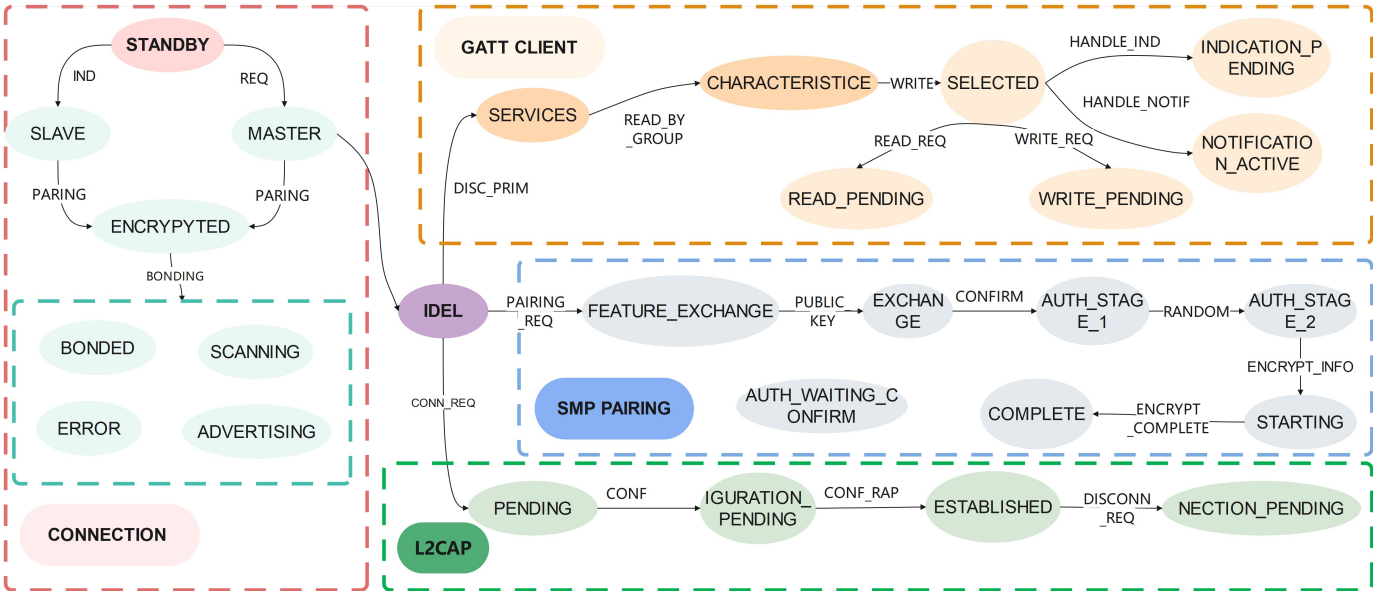

Fig. 2. State machine model.

into a 7-dimensional vector input to the DQN strategy selector, which employs an ϵ-greedy policy to select one of five semantic strategies balancing exploration and exploitation. Third, the Semantic Variation Engine executes the chosen strategy through either havoc mutation (AFL-style random bit-flipping) or state-driven mutation (querying state transition tables). Fourth, mutated packets are transmitted to the Target with 2-second timeout protection. Fifth, execution outcomes are quantified as reward signals using (1), considering coverage increments, timeouts, and invalid transitions, with experience tuples stored in the replay buffer.

The two-stage training mechanism operates concurrently with the fuzzing loop. In phase 1, expert demonstrations are recorded by logging state-strategy pairs to CSV files. In phase 2, offline training periodically uses both expert demonstrations for supervised learning and replay buffer experiences for deep Q-learning, updating network parameters through gradient descent and synchronizing target networks every 100 steps.

These components work synergistically in a closed-loop architecture to enable state-aware adaptive protocol fuzzing where all decisions derive from the current 7-dimensional protocol state. The DQN learns optimal strategy selection through historical experience, while semantic-driven mutations leverage or violate protocol specifications based on selected strategies rather than arbitrary bit-flipping.

A. Protocol Modeling

We formulate BLE fuzzing as a MDP where the fuzzer dynamically selects mutation strategies based on protocol state to maximize cumulative coverage rewards. The MDP comprises a 7-dimensional state space S, five semantic actions A, and a reward function R defined in Equation 1. The BLE protocol is modeled as a four-layer state machine with 44 states across LL, GATT, SMP, and L2CAP layers as shown in Fig. 2, providing granularity for strategy selection while maintaining computational tractability.

These states, defined in the Bluetooth Core Specification [5], encompass connection establishment, service discovery, security pairing, and data transfer phases. Although the complete BLE state space includes additional implementation-specific states, our model captures protocol-critical states. This layered design aligns with the BLE stack architecture, where each layer manages independent functionalities requiring distinct fuzzing strategies. The state space S is encoded as a 7-dimensional vector balancing representational capacity with learning efficiency.

The first four dimensions capture discrete protocol states for LL, GATT, SMP, and L2CAP layers respectively, enabling layer-specific context distinction. The remaining three dimensions encode triggered state transitions, recent invalid packet rate, and coverage increment, directly reflecting fuzzing effectiveness and guiding strategy adaptation. Following prior RL-based fuzzing work [30], the action space $\mathcal{A}$ comprises five semantic mutation strategies detailed in Section III-B, and the reward function $\mathcal{R}$ is defined as:

$$\mathcal{R} = 0.6 \cdot \Delta_{\text{cov}} - 0.2 \cdot r_{\text{timeout}} - 0.15 \cdot r_{\text{invalid}} + 0.05 \cdot n_{\text{transition}} \quad (1)$$

where coefficients denote relative importance: Δ_{cov} represents coverage increment as the primary reward; r_{timeout} and r_{invalid} penalize timeouts and invalid packets respectively to reduce wasted fuzzing cycles; and $n_{\text{transition}}$ counts new state transitions to encourage exploration of deep protocol states.

B. Semantic Variation Engine

Traditional mutation-based fuzzers employ blind bit-flipping or byte-level modifications devoid of protocol semantics. This approach generates high proportions of invalid packets that

rarely reach deep protocol states in upper layers such as GATT and SMP.

We design five semantic mutation strategies grounded in systematic analysis of the Bluetooth Core Specification [5], which adaptively switch between protocol layers and test targets through constraint-driven mutation logic:

- CONFORMANCE generates valid test cases within protocol specification bounds, modifying non-critical fields to explore normal execution paths while ensuring compliance through strict constraint checking.
- BOUNDARY performs boundary value analysis on fields with range or length constraints, testing edge cases at protocol-defined limits.
- TRANSITION targets unexplored state transitions by querying the state transition table containing up to 128 rules, selecting PDUs leading to unvisited states to rapidly cover deep states in GATT and SMP layers.
- VIOLATION breaches protocol constraints using illegal opcodes, excessively long fields, or invalid state transitions to identify input validation vulnerabilities and DoS flaws.
- INTELLIGENT combines all aforementioned mutation methods through multi-field coordination and sequence variation to discover complex vulnerabilities and timing-related defects.

The Semantic Variation Engine dynamically selects packets, fields, and strategies through constraint-driven mutation logic. This logic considers three factors: constraint types (e.g., range constraints, opcode validity), state relevance (alignment with current protocol layer), and historical mutation counts (avoiding redundant mutations).

C. RL-Enhanced Strategy Selector

Unlike fixed-strategy fuzzers that apply uniform mutation policies regardless of protocol state, our method employs RL to dynamically select optimal strategies through iterative interaction. We implement this adaptive capability via a DQN with a three-layer fully connected architecture (7-64-32-5) designed to balance representational expressiveness and generalization. The input layer accepts a normalized 7-dimensional protocol state vector, two hidden layers (64 and 32 neurons with ReLU activation) extract latent features, and the output layer produces Q-values for five semantic mutation strategies. The module follows a two-stage training paradigm combining offline supervised pre-training with online RL.

Phase 1: Offline Supervised Learning. This phase extracts state sequences from historical fuzzing logs and synthesizes training labels using heuristic rules that map protocol states to appropriate strategies. For instance, states with high invalid packet rates are labeled with VIOLATION to test robustness, long-lived states are labeled with TRANSITION to encourage exploration, and GATT discovery phases are labeled with TRANSITION to explore deep protocol states. The model is trained using cross-entropy loss for 100 epochs, and the resulting pre-trained weights initialize the online DQN, providing a warm start that mitigates the cold-start problem.

Phase 2: Online Reinforcement Learning. The DQN interacts with the BLE target using experience replay and an ϵ-greedy policy. At each decision step, the agent either explores by randomly selecting a strategy or exploits by choosing the strategy with the highest Q-value. Transitions (s, a, r, s') are stored in the replay buffer, with mini-batches periodically sampled for gradient-based updates to the online network using the Adam optimizer. A separate target network provides stable Q-value estimates through periodic synchronization. The exploration rate ϵ decays from 0.3 to 0.05 to balance exploration and exploitation.

This two-stage framework effectively combines supervised and RL advantages. Phase 1 establishes baseline strategy-selection patterns through heuristic rules, providing the online DQN with informed initial weights. Phase 2 then adapts to protocol-specific characteristics through interactive learning while retaining RL's adaptive capability. This approach yields significantly better initial performance and faster convergence compared to training from scratch.

IV. Evaluation

To evaluate the effectiveness of RL-Semantic, we designed and conducted several comprehensive experiments on the following research questions:

RQ1: Does RLsemantic achieve superior overall performance compared to baseline methods in terms of state transition coverage and efficiency?

RQ2: Can RLsemantic achieve comprehensive coverage across all BLE protocol layers, particularly for security-critical layers such as SMP and GATT?

RQ3: What are the individual contributions of semantic strategies and RL-enhanced adaptive selection to the overall performance improvement?

A. Experimental setup

We conduct experiments in the BLuEMan fuzzing environment with comprehensive protocol coverage and reproducible experimental settings. Each target runs for 360 minutes repeated 3 times with averaged results reported, where all experiments execute on identical hardware with Docker environment isolation to ensure reproducibility.

a) Dataset: We use the official BLuEMan benchmark dataset [4] comprising 10 real-world BLE implementations from open-source RTOS projects including Apache Mynewt, Zephyr, and NimBLE. These targets span four protocol configurations (GATT operations, heart rate monitoring, SMP pairing, L2CAP credit flow control) with both central/peripheral or client/server roles, comprehensively covering all four BLE protocol stack layers (LL, GATT, SMP, L2CAP). Each implementation is compiled as a single ELF executable enabling systematic fuzzing in a simulation environment.

b) Baseline Methods: We compare RL-Semantic against three baselines. Field represents state-of-the-art protocol-aware fuzzing using random field modification without semantic guidance. AFL applies traditional havoc mutation strategies treating BLE packets as raw byte sequences without protocol

TABLE I
STATE TRANSITION PERFORMANCE COMPARISON (360-MINUTE TEST)

Target	Mutator							
	Field		AFL		Random		RL-Semantic	
	Count	STC	Count	STC	Count	STC	Count	STC
gatt_write_peripheral	64	0.025	64	0.090	64	0.063	275	0.107
hr_peripheral	64	0.024	64	0.079	64	0.153	273	0.076
sm_pairing_peripheral	18	1.050	37	3.140	54	2.418	77	**7.653**
lc_credit_server	63	0.020	64	0.080	64	0.152	**281**	0.075
ots_peripheral	64	0.009	64	0.048	64	0.134	259	0.028
gatt_write_central	48	0.077	60	0.125	64	0.136	202	0.067
hr_central	64	0.007	64	0.057	64	0.135	279	0.026
sm_pairing_central	62	0.021	64	0.089	64	0.132	**281**	0.025
lc_credit_client	61	0.005	64	0.070	64	0.135	279	0.016
otc_central	43	0.005	64	0.059	64	0.133	**280**	0.025
Central Avg	56	0.023	63	0.080	64	0.134	264	0.032
Overall Avg	55	0.023	61	0.093	63	0.134	266	0.061

structure awareness. Random employs completely random byte-wise mutation as a minimal baseline to demonstrate the necessity of protocol-aware fuzzing.

c) Evaluation Metrics: We use three metrics to evaluate fuzzing performance: State Transition Coverage (STC), Absolute State Transition Count, and Protocol Layer Coverage.

State Transition Coverage (STC) measures exploration efficiency and is defined as:

$$STC = \frac{N_{\text{transition}}}{N_{\text{total}}} \times 1000 \tag{2}$$

where $N_{\text{transition}}$ is observed state transitions and N_{total} is total packets sent. Higher STC indicates more efficient state-space exploration and eliminates packet count bias, making it suitable for comparing semantic-aware against packet-only methods.

Absolute State Transition Count records the total number of unique state transitions triggered during fuzzing. This metric is critical because many protocol vulnerabilities depend on specific state transition sequences. Fig. 3 illustrates this using the BLESA attack [31], which requires the device to transition from Bonded to Reconnecting states while bypassing identity verification. Such state-dependent behavior means fuzzers must systematically trigger complete transition paths, which packet-only mutation cannot achieve without maintaining protocol state context.

Protocol Layer Coverage quantifies the number of successfully triggered state transitions across GATT, SMP, L2CAP, and LL layers, measuring deep protocol stack exploration capability.

B. Overall Performance Comparison

To answer **RQ1**, we compare RL-Semantic against three baselines (Field, AFL, and Random) across ten BLE targets over 360-minute fuzzing campaigns.

RL-Semantic demonstrates substantial performance advantages, achieving a 4.8 times improvement in absolute state transitions and 2.7 times improvement in STC efficiency. As shown in Table I, RL-Semantic achieves 266 average transitions versus 55 for Field, 61 for AFL, and 63 for Random. Field consistently bottlenecks at approximately 64 transitions on six targets, with individual experiments reaching 65, confirming this limitation stems from Field's methodology rather than experimental artifacts. The most significant improvements occur on SMP layer targets, where RL-Semantic achieves 77 and 281 transitions compared to Field's 18 and 62.

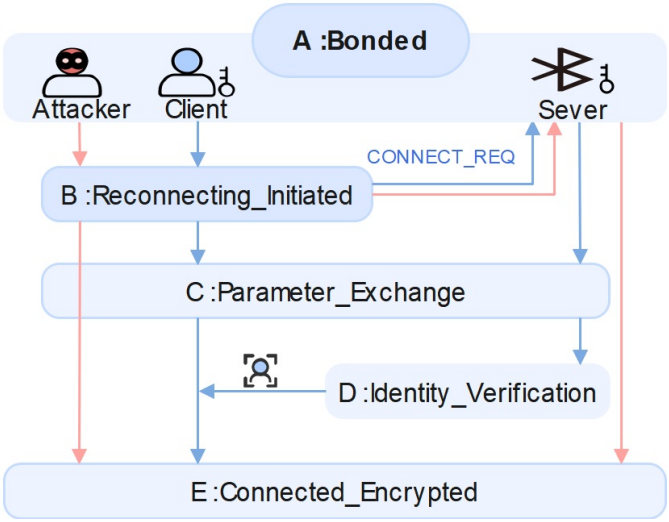

Fig. 3. BLESA vulnerability attack state transition.

RL-Semantic's superior performance stems from three technical innovations. First, while Field's random modifications produce invalid packets rejected by the protocol stack, RL-Semantic's TRANSITION strategy queries a 44-state hierarchical model for valid transitions and VIOLATION strategy tests error paths. Second, the DQN component enables efficient resource allocation by prioritizing strategies based on protocol

TABLE II
PROTOCOL LAYER COVERAGE COMPARISON

Protocol Layer	Field	RL-Semantic
GATT	909	6,251
SMP	0	139,494
L2CAP	1,796	5,505
LL	313,561	423,169

state, whereas Field applies uniform mutations regardless of context. Third, semantic guidance for security-critical layers enables breakthrough performance where Field cannot maintain precise SMP pairing sequences, as demonstrated by superior transitions on targets with higher STC values. These innovations confirm that semantic strategies maintain protocol validity while maximizing exploration across diverse scenarios.

C. Protocol Layer Coverage

To investigate research question **RQ2**, we analyze protocol layer coverage by counting successfully triggered state transitions at each protocol layer. This analysis reveals whether RL-Semantic can comprehensively explore deep protocol stack states, particularly in security-critical layers. RL-Semantic demonstrates decisive advantages at GATT and SMP layers while achieving modest improvements at LL and L2CAP layers as shown in Table II.

RL-Semantic achieves 6.9 times the coverage at GATT and transitions from zero to non-zero coverage at SMP where Field completely fails to trigger any state transitions. This failure occurs because Field's random modifications cannot navigate SMP's complex pairing state machine requiring precise opcode sequences, whereas RL-Semantic's TRANSITION strategy explicitly targets these deep states while VIOLATION strategy tests error handling paths. The substantial GATT improvement stems from BOUNDARY strategy systematically testing boundary values and CONFORMANCE strategy ensuring valid operations, whereas Field is uniform mutations frequently generate invalid opcodes rejected early by the protocol stack. Conversely, improvements at LL and L2CAP layers are relatively modest because LL handles fundamental operations inherently triggered at high frequency across all methods, while L2CAP operations are simple enough for Field to achieve partial coverage through high-frequency mutation. This validates that semantic strategies provide the greatest value for complex state-dependent protocol layers.

D. Ablation Study

To investigate **RQ3**, we conduct ablation experiments to verify the contributions of each component. We compare Field as baseline against Semantic Strategies incorporating five semantic mutation approaches, and RL-Semantic Selection adding DQN adaptive selection. Fig. 4 presents the incremental contributions of each component across ten BLE targets.

a) Semantic Strategies vs Field: Semantic strategy achieves substantial improvement over Field method, representing a 4.5 times enhancement. This provides empirical validation of protocol-aware semantic strategy efficacy, as Semantic contributes the majority of state transitions on SMP layer targets. These findings demonstrate that constraint-guided strategies generate higher-fidelity test cases, facilitating deeper protocol-relevant state transitions.

b) RL-Semantic vs Semantic: RL adaptive strategy selection provides additional improvement, with RL-Semantic achieving approximately 257 average transitions compared to 246 for Semantic. The modest gains demonstrate the feasibility of RL in protocol fuzzing, validating that RL can optimize strategy selection even when baseline semantic strategies already achieve strong performance.

c) Ablation Summary: The dominant contribution of Semantic over RL enhancement confirms that protocol constraints are more critical than adaptive selection, yet RL demonstrates potential for further optimization.

d) Discussion on RL Effectiveness: The modest improvement from RL enhancement stems from two factors. First, the current DQN-based approach represents a feasibility demonstration using basic RL architecture; more sophisticated algorithms such as Policy Gradient or Actor-Critic could potentially yield greater optimization. Second, the extended six-hour testing duration enables both RL-enhanced and non-RL approaches to converge toward high coverage states, diminishing the relative advantage of adaptive selection over time. This suggests that RL demonstrates effectiveness primarily during early exploration phases where adaptive strategy selection accelerates state space navigation.

V. CONCLUSION

In this paper, we present RL-Semantic, which integrates protocol-aware semantic mutation strategies with RL for adaptive policy selection. The approach models the BLE protocol stack as a four-layer hierarchical state machine and formulates the fuzzing process as an MDP, implementing five semantic mutation strategies through a DQN-based adaptive selector trained via a two-stage approach combining offline supervised learning and online RL.

Semantic strategies constitute the primary performance driver, achieving a 4.8 times improvement over the Field baseline, while RL provides additional optimization through dynamic strategy selection. Evaluated across ten emulated BLE targets, RL-Semantic achieves an average of 266 state transitions compared to 55 for Field. The framework demonstrates substantial advantages at security-critical protocol layers, achieving a 6.9 times improvement at the GATT layer and successfully enabling non-zero coverage at the SMP layer.

REFERENCES

[1] D.-P. Schreiber, M. Leithner, J. Zivanovic, and D. E. Simos, "Bluetooth low energy security testing with combinatorial methods," in *2025 USENIX Annual Technical Conference (USENIX ATC 25)*, pp. 1625–1638, 2025.

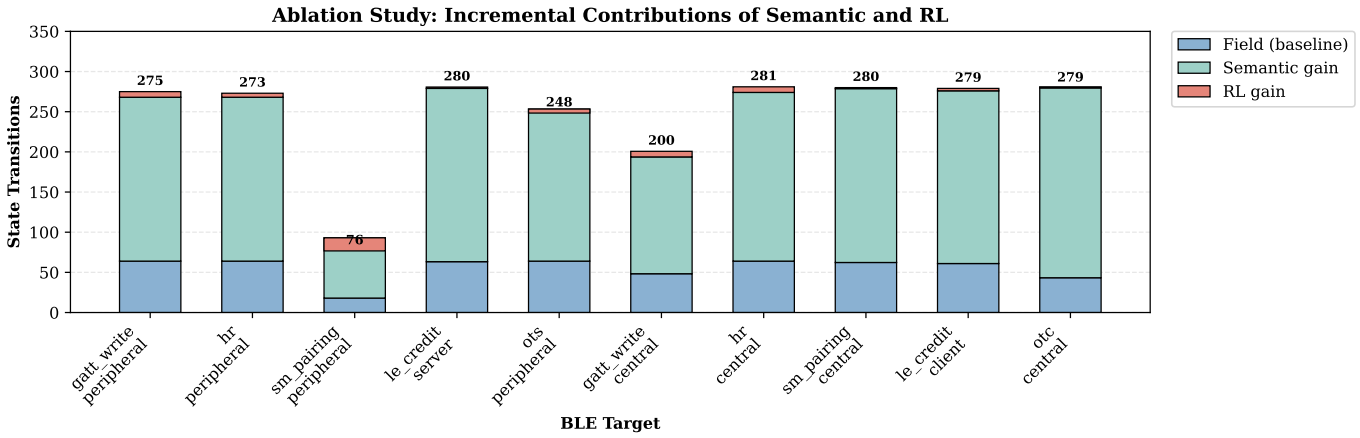

Fig. 4. Incremental contributions of Field, Semantic, and RL-Semantic across ten BLE targets.

[2] C. Skallak and S. Schmidt, "Indescribably blue: Bluetooth low energy threat landscape.," in *IoTBDS*, pp. 339–350, 2024.

[3] Z. Wang, "Securing bluetooth low energy: A literature review," *arXiv preprint arXiv:2404.16846*, 2024.

[4] W.-C. Kao, Y.-C. Chen, Y.-S. Lin, Y.-C. Yang, C.-Y. Li, and C.-Y. Huang, "{BLuEMan}: A stateful simulation-based fuzzing framework for {Open-Source}{RTOS} bluetooth low energy protocol stacks," in *34th USENIX Security Symposium (USENIX Security 25)*, pp. 6259–6278, 2025.

[5] B. S. I. Group, *Bluetooth Core Specification v5.4*. Bluetooth SIG, 2023. https://www.bluetooth.com/specifications/bluetooth-core-specification/.

[6] X. Zhang, C. Zhang, X. Li, Z. Du, B. Mao, Y. Li, Y. Zheng, Y. Li, L. Pan, Y. Liu, *et al.*, "A survey of protocol fuzzing," *ACM Computing Surveys*, vol. 57, no. 2, pp. 1–36, 2024.

[7] M. Zalewski, "Fuzzing with code coverage," in *USENIX Security*, pp. 437–454, 2014.

[8] A. Fioraldi, D. Maier, H. Eißfeldt, and M. Heuse, "{AFL++}: Combining incremental steps of fuzzing research," in *14th USENIX workshop on offensive technologies (WOOT 20)*, 2020.

[9] R. Meng, V.-T. Pham, M. Böhme, and A. Roychoudhury, "Aflnet five years later: On coverage-guided protocol fuzzing," *IEEE Transactions on Software Engineering*, 2025.

[10] J. Qiu, Y. Jiang, Y. Miao, W. Luo, L. Pan, and X. Zheng, "A survey of coverage-guided greybox fuzzing with deep neural models," *Information and Software Technology*, p. 107797, 2025.

[11] J. Jung, P. Hosek, T. Sugawara, *et al.*, "Aflsmart: Combining mutation and fuzzing," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1300–1311, 2020.

[12] M. Böhme, V.-T. Pham, and M.-D. Nguyen, "Directed greybox fuzzing," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2329–2344, 2017.

[13] F. Detto, N. Asokan, and S. Niebuhr, "Mcfluffin: Fuzzing the bluetooth controller via state machine model inference," in *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 2813–2830, 2023.

[14] F. Wu, Z. Luo, Y. Zhao, Q. Du, J. Yu, R. Peng, H. Shi, and Y. Jiang, "Logos: Log guided fuzzing for protocol implementations," in *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 1720–1732, 2024.

[15] X. Song, Y. Zeng, J. Wu, H. Li, C. Zuo, Q. Zhao, and S. Guo, "Csfuzzer: A grey-box fuzzer for network protocol using context-aware state feedback," *Computers & Security*, p. 104581, 2025.

[16] Y. Zhang, K. Zhu, J. Peng, Y. Lu, Q. Chen, and Z. Li, "Statepre: A large language model-based state-handling method for network protocol fuzzing," *Electronics*, vol. 14, no. 10, p. 1931, 2025.

[17] K. Jian, Y. Zou, Y. Li, J. Cao, M. Li, J. Sun, J. Shi, and W. Huo, "Fuzzing for stateful protocol implementations: Are we there yet?," in *International Symposium on Theoretical Aspects of Software Engineering*, pp. 186–204, Springer, 2024.

[18] K. Jian, Y. Zou, M. Li, and W. Huo, "Fuzzing for stateful protocol programs based on constraints between states and message types," in *Proceedings of the 16th International Conference on Internetware*, pp. 305–316, 2025.

[19] C. Cadar and K. Sen, "Symbolic execution for software testing: Three decades later," *Communications of the ACM*, vol. 56, no. 2, pp. 82–90, 2013.

[20] Q. Wang, B. Chang, S. Ji, Y. Tian, X. Zhang, B. Zhao, G. Pan, C. Lyu, M. Payer, W. Wang, *et al.*, "Syztrust: State-aware fuzzing on trusted os designed for iot devices," in *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 2310–2387, IEEE, 2024.

[21] P. R. Chaudhary and R. R. Maiti, "Iotfuzzsentry: A protocol guided mutation based fuzzer for automatic vulnerability testing in commercial iot devices," *arXiv preprint arXiv:2509.09158*, 2025.

[22] X. Ma, L. Luo, and Q. Zeng, "From one thousand pages of specification to unveiling hidden bugs: Large language model assisted fuzzing of matter {IoT} devices," in *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 4783–4800, 2024.

[23] M. Maugeri, "Fuzzing matter (s): A white paper for fuzzing the matter protocol.," in *ICISSP*, pp. 446–451, 2024.

[24] J. Su, H.-N. Dai, L. Zhao, Z. Zheng, and X. Luo, "Effectively generating vulnerable transaction sequences in smart contracts with reinforcement learning-guided fuzzing," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pp. 1–12, 2022.

[25] M. Foley and S. Maffeis, "Apirl: Deep reinforcement learning for rest api fuzzing," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, pp. 191–199, 2025.

[26] M. Sablotny, B. S. Jensen, and J. Singer, "Reinforcement learning guided fuzz testing for a browser's html rendering engine," *arXiv preprint arXiv:2307.14556*, 2023.

[27] X. Wan, T. Li, W. Lin, Y. Cai, and Z. Zheng, "Coverage-guided fuzzing for deep reinforcement learning systems," *Journal of Systems and Software*, vol. 210, p. 111963, 2024.

[28] C. Tao, Y. Tao, H. Guo, Z. Huang, and X. Sun, "Dlregion: Coverage-guided fuzz testing of deep neural networks with region-based neuron selection strategies," *Information and Software Technology*, vol. 162, p. 107266, 2023.

[29] W. Lin, Y. Xie, R. Liu, Z. Dang, J. Hao, T. Li, Z. Ma, and J. Ma, "Deep2fuzz: Coverage-guided reinforcement fuzzing towards deep neural networks," in *2025 International Conference on Networking and Network Applications (NaNA)*, pp. 258–263, IEEE, 2025.

[30] Y. Shen, Y. Liu, and Y. Zhou, "Rlgfuzz: Reinforcement learning guided fuzzing with state-coverage mapping environment," in *2024 IEEE 32nd International Conference on Network Protocols (ICNP)*, pp. 1–9, IEEE, 2024.

[31] J. Wu, Y. Nan, V. Kumar, D. J. Tian, A. Bianchi, M. Payer, and D. Xu, "{BLESA}: Spoofing attacks against reconnections in bluetooth low energy," in *14th USENIX Workshop on Offensive Technologies (WOOT 20)*, 2020.