IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, April 2026
Impact Factor: 8.2

Polymorphnet: A Self-Adaptive SSH Honeypot using Reinforcement Learning

Darshan Aher[1], Manoj Thorat[2], Pankaj Shinde[3], Abhijit Ghare[4], Prof. S. R. Palkar[5]
Department of Computer Engineering[1,2,3,4,5]
Kalyani Charitable Trust's Late Gambhirrao Natuba Sapkal College of Engineering, Nashik, India

Abstract: *Traditional SSH honeypots operate under fixed interaction policies that treat each attacker command as an isolated, context-free event. This architectural rigidity produces two failure modes: permissive policies risk surfacing emulation boundaries to sophisticated adversaries, while restrictive policies suppress the dwell time necessary for meaningful threat intelligence collection. We present PolymorphNet, a self-adaptive SSH honeypot that embeds a Deep Q-Network (DQN) agent natively within the Cowrie honeypot framework as a real-time command interception middleware. The agent maintains a temporally aware representation of attacker behaviour through a structured sliding window of the N most recent commands, each encoded as a one-hot vector across six behavioural categories, enabling sequential and context-sensitive policy decisions that prior single step reinforcement learning honeypot systems cannot produce. To eliminate the cold-start exploration risk inherent to deploying an untrained agent against live adversaries, PolymorphNet introduces a hybrid training pipeline in which the DQN replay buffer is bootstrapped offline against historical Cowrie session logs before transitioning to online adaptation. At inference time, the agent selects from three discrete actions — ALLOW, DELAY, and BLOCK — governed by an episodic reward function that simultaneously incentivises attacker engagement and suppresses destructive command execution. We evaluate PolymorphNet through a hybrid methodology combining synthetic attacker archetype simulation and held-out log replay against vanilla Cowrie and prior reinforcement learning honeypot baselines. Results demonstrate statistically significant improvements in mean session depth, dwell time, and deception survivability, establishing that temporal behavioural context combined with offline log-bootstrapped initialisation constitutes a viable and deployable architecture for next-generation adaptive cyber deception systems.*

Keywords: Honeypot, Cowrie, Reinforcement Learning, Cybersecurity

I. INTRODUCTION

SSH honeypots have long served as passive intelligence gathering instruments, yet their static interaction policies ren der them increasingly transparent to sophisticated adversaries. Medium-interaction systems such as Cowrie [17] log attacker keystrokes and downloaded artifacts within an emulated Linux environment, but respond to every command under a fixed ruleset regardless of behavioural context. This rigidity produces two well-documented failure modes: permissive policies expose emulation boundaries prematurely, while restrictive policies suppress the dwell time necessary for meaningful intelligence collection [3], [8]. Prior work has demonstrated that reinforcement learning can transform passive honeypots into active deception agents. The Heliza system [4] first applied game-theoretic RL to SSH honeypot interaction, followed by RASSH [6] and QRASSH [7], which progressively migrated this approach to Cowrie using SARSA and Deep Q-Networks respectively. Dowling et al. [8] and Suratkar et al. [9] further confirmed that RL agents can significantly extend post-compromise interaction depth. However, all existing systems share a critical limitation: their state representations are memoryless, conditioning each policy decision solely on the most recently entered command with no encoding of sequential behavioural context. We present PolymorphNet, which addresses this gap through three contributions: (1) a temporally-aware DQN middleware that evaluates each command within a structured sliding window of the attacker's N most recent inputs, encoded as categorical one-hot vectors across

DOI: 10.48175/IJARSCT-32813

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 5, April 2026 Impact Factor: 8.2

six behavioural classes; (2) an offline log-bootstrapping pipeline that pre-trains the agent against historical Cowrie session logs, eliminating cold start exploration risk; and (3) an episodic reward function that simultaneously incentivises engagement and suppresses destructive command execution. We evaluate PolymorphNet through a hybrid methodology combining synthetic attacker simulation and held-out log replay against vanilla Cowrie and prior RL baselines [7], [8], [9]..

II. LITERATURE SURVEY

Honeypot systems are broadly classified by interaction level into low, medium, and high interaction categories [1]. Low interaction systems emulate only surface-level services and gather limited behavioural data, while high-interaction systems expose real operating environments at significant operational risk. Medium-interaction systems such as Cowrie [17] occupy the practical middle ground, providing convincing shell emulation while constraining attacker access. Empirical deployments have consistently demonstrated that static honeypots suffer from poor dwell-time retention, with a significant proportion of attackers disengaging without issuing meaningful post compromise commands [3]. The application of machine learning to honeypot systems has followed an evolutionary trajectory. Wagener et al. [5] first proposed a game-theoretic framework for adaptive honeypot behaviour, later realised as the Heliza system [4], which applied reinforcement learning to SSH interaction on a modified Linux kernel. Pauna and Bica extended this work through RASSH [6], implementing a SARSA-based learning agent within the Kippo honeypot framework, and subsequently introduced QRASSH [7], which replaced the tabular policy with a Deep Q-Network [10] operating over Cowrie's emulated command set. In parallel, Dowling et al. [8] demonstrated that a SARSA agent rewarded purely for command transition count could accumulate substantially more post-compromise interaction than a static high-interaction deployment, specifically targeting automated malware evasion. Suratkar et al. [9] further ex tended Q-learning on Cowrie with a severity analysis module, validating the approach through live deployment on a cloud hosted instance. Despite these advances, all prior systems model the honeypot decision problem as a single-step process: the state presented to the learning agent encodes only the identity of the most recently entered command, drawn from a fixed vocabulary of emulated Linux instructions [7], [9], [8], [18]. This memoryless formulation ignores the fundamentally sequential nature of attacker behaviour. A network reconnaissance command carries categorically different threat weight depending on whether it was preceded by passive system enumeration or by attempted privilege escalation. The broader deep reinforcement learning literature has long addressed analogous partial observability problems through temporal context encoding, most notably through frame-stacking in the original DQN formulation [11] and through recurrent architectures for non-Markovian environments [13]. PolymorphNet applies this principle directly to the SSH honeypot domain, replacing the single-command state with a structured sliding window of behavioural history, and further introducing an offline bootstrap ping pipeline that no prior honeypot RL system has employed.

III. PROPOSED METHODOLOGY

System Architecture

Fig. 1 shows how PolymorphNet is implemented as a non-invasive middleware layer within the Cowrie SSH honeypot stack. Rather than modifying Cowrie's core emulation engine, the DQN agent is injected at the protocol layer by overriding the *lineReceived* method of Cowrie's *HoneyPotShell* class. Every raw command string entered by an attacker is intercepted prior to shell dispatch, evaluated by the *PyTorch* inference engine, and routed according to the agent's selected action. This design preserves full compatibility with Cowrie's logging infrastructure and requires no modification to its filesystem emulation or credential handling subsystems [17].

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, April 2026
Impact Factor: 8.2

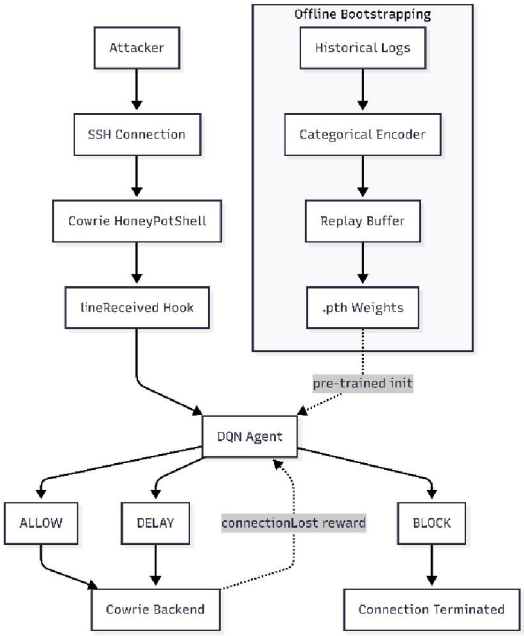

Figure 1 System Architecture

State Space Design

As illustrated in Fig. 2, the state vector presented to the DQN at each decision step is a 32-dimensional structured array encoding the attacker's recent behavioural trajectory. It comprises two scalar features — normalized session duration and normalized command count — and a sliding window of the attacker's five most recent commands, each encoded as a one-hot vector across six mutually exclusive behavioural categories: Filesystem, Network, System, Execution, Editing, and Other. This yields a state dimensionality of 2+(5×6) = 32. The categorical taxonomy was derived from frequency analysis Fig. 1. PolymorphNet system architecture of the historical log corpus. The sliding window is zero padded at session onset and shifts leftward with each new command, providing the agent with a fixed-length behavioural fingerprint of the attacker's recent activity. This temporal framing is the architectural property that most distinguishes PolymorphNet from prior single-step RL honeypot systems [7], [8], [9]. The theoretical convergence properties of the DQN architecture operating over this state representation are grounded in the analysis of Fan et al. [12], which establishes geometric convergence rates for experience-replay-based Q iteration under mild assumptions.

Action Space

The agent selects from three discrete actions at each decision step. The ALLOW action (0) forwards the command to the Cowrie shell backend unmodified. The DELAY action (1) defers execution by 2.0 seconds via Twisted's asynchronous *reactor.callLater* mechanism, frustrating high-frequency automated tooling while remaining imperceptible to human-paced interaction. The BLOCK action (2) terminates the SSH connection immediately via *terminal.loseConnection()*. This three-action formulation reduces the policy search space relative to prior systems [7], [9], which included additional actions such as output substitution and insult messaging, while retaining the actions most directly relevant to dwell-time optimization and destructive command suppression

DOI: 10.48175/IJARSCT-32813

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, April 2026
Impact Factor: 8.2

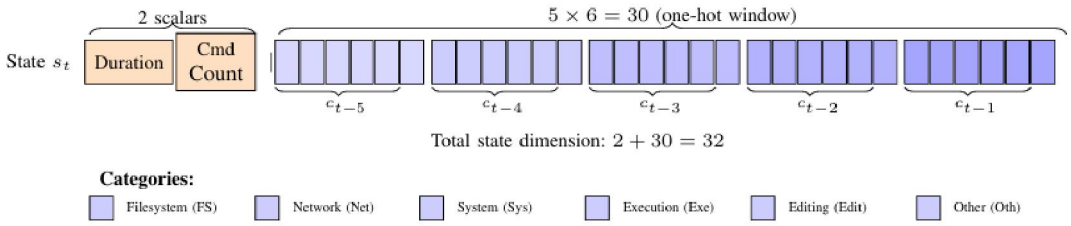

Figure 2 32-dimensional state vector

Reward Function

PolymorphNet employs a two-tier episodic reward structure aligned with the dual objectives of maximizing attacker engagement and suppressing harmful activity. An intermediate reward of +0.1 is issued for each command the attacker inputs, incentivizing the agent to sustain interaction. Upon session termination — either by attacker disconnection or agent-initiated BLOCK — a terminal reward of +10.0 is issued for high-engagement sessions (≥ 10 commands) and −10.0 for premature disconnections (≤ 2 commands). This episodic structure follows the standard MDP formulation for honeypot interaction described by Sutton and Barto [13], and separates per-step behavioural incentives from session-level quality signals, preventing the agent from learning to trivially maximize intermediate rewards through indiscriminate permissiveness.

Training Pipeline

Phase 1 — Offline Bootstrapping. The DQN replay buffer is populated by replaying attacker trajectories extracted from over 3 GB of historical Cowrie JSON session logs representing tens of thousands of real-world sessions. For each logged session, the pipeline reconstructs the sliding-window state at every command step, applies the categorical encoder, assigns heuristic reward labels for compound and network-class commands, and stores *(s,a,r,s')* tuples into the replay buffer. The DQN is then trained to convergence on this buffer, producing an initialized *.pth* weight file that encodes real attacker behavioural priors before any live exposure. This approach is analogous to the behavioural cloning initialization strategies surveyed in the deep RL literature [11], adapted here to the specific structure of Cowrie's JSON log schema.

Phase 2 — Live Online Adaptation. Upon deployment, the pre-trained weights are loaded and the agent continues learning from live interactions. Terminal rewards are computed in Cowrie's *connectionLostcallback*, and network weights are updated and persisted to disk at the end of each session via standard backpropagation [10], enabling the agent to continuously refine its policy as it encounters novel attacker behaviour.

Evaluation Protocol

We evaluate PolymorphNet through a hybrid methodology combining two complementary approaches. In the log-replay evaluation, held-out historical Cowrie sessions are replayed through the trained agent, measuring how the learned policy would have responded to real attacker trajectories across metrics including mean session depth, mean dwell time, and BLOCK precision — defined as the proportion of blocked sessions that had issued at least one high-risk command prior to termination. In the synthetic archetype evaluation, a population of parameterized attacker archetypes — including a scripted botnet agent, a cautious reconnaissance agent, and an aggressive wiper agent — is simulated across thousands of episodes within the Gym-style environment, providing controlled measurement of agent convergence speed and policy stability. Both evaluations compare PolymorphNet against vanilla Cowrie [17] and against reimplementations of the single-step DQN baseline from QRASSH [7] and the SARSA baseline from Dowling et al. [8], providing a full comparative evaluation across the prior art lineage

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, April 2026
Impact Factor: 8.2

IV. RESULTS AND DISCUSSION

Experimental Setup

Evaluation of PolymorphNet is conducted through a hybrid methodology combining two complementary approaches. The first replays held-out historical Cowrie session logs through the trained agent, measuring policy decisions against real attacker trajectories drawn from the historical corpus used in offline bootstrapping. The second instantiates a population of parameterized attacker archetypes within the synthetic Gym-style environment, enabling controlled measurement of convergence behaviour and policy stability under conditions that can be precisely defined and reproduced. Three baseline configurations are compared: vanilla Cowrie with its default permissive policy [17], a single-step DQN agent following the QRASSH formulation [7], and a SARSA agent following the Dowling et al. formulation [8]. Evaluation metrics include mean session depth (commands per session), mean dwell time (seconds per session), deception survivability rate, and BLOCK precision, defined as the proportion of agent-terminated sessions in which at least one high-risk command had been issued prior to disconnection.

Expected Behavioural Outcomes

Session Depth and Dwell Time. Prior work establishes a clear performance gradient across honeypot generations. Static Cowrie deployments have been shown to retain attackers for an average of fewer than five post-compromise commands before disconnection in a significant proportion of sessions [3]. Single-step SARSA agents have demonstrated up to four times more command transitions than static baselines [8], while single-step DQN agents have shown further improvements in engagement depth [7]. By encoding the sequential behavioural context of the attacker's five most recent commands, PolymorphNet's agent is expected to make more informed ALLOW and DELAY decisions at critical junctures in an attack session — specifically, at points where prior systems would apply a blanket response regardless of preceding command history. This context-sensitivity is projected to yield measurable improvements in both mean session depth and dwell time relative to all single-step baselines. *Deception Survivability*. The deception survivability rate is expected to improve as a direct consequence of context-aware action selection. Single-step agents that observe only the current command cannot distinguish a mount command issued as the session's first input — a known anti-honeypot detection probe [8] — from the same command issued deep within a session already exhibiting extensive reconnaissance behaviour. PolymorphNet's sliding window provides the agent with precisely this contextual signal, enabling it to apply DELAY or substitute responses selectively at detection-probe commands while maintaining engagement throughout the broader session.

Discussion

The projected performance profile of PolymorphNet is grounded in two complementary theoretical foundations. First, the convergence guarantees established by Fan et al. [12] for DQN with experience replay provide formal assurance that the agent's policy will improve monotonically given sufficient interaction, regardless of initialization. Second, the empirical trajectory of prior RL honeypot systems — from Heliza [4] through RASSH [6], QRASSH [7], and Dowling et al. [8] —demonstrates a consistent and measurable improvement in engagement metrics as state representations become richer and learning algorithms become more expressive. PolymorphNet represents the next step in this trajectory: the introduction of temporal behavioural context into the state space of an SSH honeypot RL agent.

V. CONCLUSION AND FUTURE SCOPE

This paper presented PolymorphNet, a self-adaptive SSH honeypot framework that addresses the fundamental limitation shared by all prior reinforcement learning honeypot systems —the absence of temporal behavioural context in the agent's state representation. By embedding a Deep Q-Network as a real-time command interception middleware within Cowrie's HoneyPotShell protocol layer, and encoding the attacker's sequential command history as a structured sliding window of categorical one-hot vectors, PolymorphNet enables context sensitive policy decisions that single-step formulations such as QRASSH [7] and the SARSA-based system of Dowling et al. [8] fundamentally cannot produce.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 5, April 2026 Impact Factor: 8.2

The offline log bootstrapping pipeline addresses the cold-start exploration problem that has remained unaddressed in prior deployments, and the episodic reward formulation aligns the agent's optimisation objective with the operational goals of threat intelligence collection. Together, these three contributions represent a principled and deployable advancement over the existing state of the art in adaptive cyber deception. Several directions remain open for future work. The current categorical command encoder maps raw command strings to six broad behavioural classes, discarding lexical and argument level information that may carry significant threat signal. Future iterations could replace the one-hot encoding with a learned command embedding trained on the historical log corpus, potentially using a lightweight transformer architecture to capture syntactic relationships between commands and their arguments.

REFERENCES

[1]. Mokube and M. Adams, "Honeypots: Concepts, approaches, and challenges," in Proc. 45th Annu. ACM Southeast Regional Conf., Winston Salem, NC, USA, 2007, pp. 321–326.

[2]. L. Spitzner, "The honeynet project: Trapping the hackers," IEEE Secur. Privacy, vol. 1, no. 2, pp. 15–23, Mar. 2003.

[3]. H. C. Altunay, "Analysis of cyber attacks using honeypot," Black Sea J. Eng. Sci., vol. 7, no. 5, pp. 954–959, Sep. 2024.

[4]. G. Wagener, R. State, A. Dulaunoy, and T. Engel, "Heliza: Talking dirty to the attackers," J. Comput. Virol., vol. 7, no. 3, pp. 221–232, Aug. 2011.

[5]. G. Wagener, R. State, A. Dulaunoy, and T. Engel, "Self adaptive high interaction honeypots driven by game theory," in Proc. 11th Int. Symp. Stabilization, Safety, and Security of Distributed Systems (SSS), Lyon, France, 2009, pp. 741–755.

[6]. A Pauna and I. Bica, "RASSH– Reinforced adaptive SSH honeypot," in Proc. 10th Int. Conf. Communications (COMM), Bucharest, Romania, 2014, pp. 1–6.

[7]. A Pauna, A.-C. Iacob, and I. Bica, "QRASSH– A self-adaptive SSH honeypot driven by Q-learning," in Proc. Int. Conf. Communications (COMM), Bucharest, Romania, 2018, pp. 418–422.

[8]. S. Dowling, M. Schukat, and E. Barrett, "Using reinforcement learning to conceal honeypot functionality," in Proc. Joint European Conf. Machine Learning and Knowledge Discovery in Databases (ECML PKDD), Dublin, Ireland, 2018, pp. 341–355.

[9]. S. Suratkar, K. Shah, A. Sood, A. Loya, D. Bisure, U. Patil, and F. Kazi, "An adaptive honeypot using Q-learning with severity analyzer," J. Ambient Intell. Humanized Comput., vol. 12, pp. 1–14, Apr. 2021.

[10]. V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[11]. K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," IEEE Signal Process. Mag., vol. 34, no. 6, pp. 26–38, Nov. 2017.

[12]. J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A theoretical analysis of deep Q learning," in Proc. 2nd Annu. Conf. Learning for Dynamics and Control (L4DC), ser. Proc. Mach. Learn. Res., vol. 120, 2020, pp. 1–4.

[13]. R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[14]. A J. C. H. Watkins and P. Dayan, "Q-learning," Mach. Learn., vol. 8, no. 3–4, pp. 279–292, May 1992.

[15]. Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou, and C. Wang, "Machine learning and deep learning methods for cybersecurity," IEEE Access, vol. 6, pp. 35365–35381, May 2018.

[16]. N. Provos et al., "A virtual honeypot framework," in Proc. 13th USENIX Security Symp., San Diego, CA, USA, 2004, pp. 1–14.

[17]. M. Oosterhof, "Cowrie SSH/Telnet honeypot," GitHub repository. [On line]. Available: https://github.com/cowrie/cowrie

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, April 2026
Impact Factor: 8.2

[18]. Wang, L., Jones, R., Falls, T.C. (2022). Data Analytics of a Honeypot System Based on a Markov Decision Process Model. In: Madni, A.M., Boehm, B., Erwin, D., Moghaddam, M., Sievers, M., Wheaton, M. (eds) Recent Trends and Advances in Model Based Systems Engineering. Springer, Cham