ARES: **A**daptive **R**einforcement Agents for **E**volving Cyber Threat**s**

Syed Ali Haider Forrest Leung
Dartmouth College
{syed.ali.haider.gr, forrest.leung.gr}@dartmouth.edu

November 10, 2025

Abstract

Traditional signature-based cybersecurity is reactive and brittle, unable to counter adaptive AI-driven attacks. We propose a hybrid GNN + MARL framework for real-time defense: GNNs encode control flow and dependencies, providing rich state representations for MARL agents that coordinate to detect, contain, and neutralize threats. An LLM-in-the-loop synthesizes, sandbox-tests, and deploys patches under MARL guidance. This closed-loop system enables continuous detection, containment, and autonomous remediation.

1 Introduction

Modern critical infrastructure, spanning finance, energy, healthcare, and defense, depends on vast software ecosystems that are increasingly targeted by adaptive and AI-driven attacks. As adversaries evolve toward autonomous, self-morphing exploits, traditional cybersecurity defenses have been rendered reactive and fragile. These systems, built on static analysis and supervised machine learning, are locked into a retrospective paradigm. They are effective at detecting known signatures but fail catastrophically when faced with unseen, zero-day threats or novel adversarial techniques designed to bypass them. The literature shows that even robust static and dynamic analysis methods have significant, exploitable trade-offs.

The clear limitations of static defenses demand a paradigm shift, a move from passive detection to active, adaptive systems that can learn, coordinate, and counter in real time. Reinforcement Learning (RL) has emerged as the leading candidate for enabling this shift. The autonomous, adaptive nature of RL is uniquely suited to the dynamic, uncertain, and sequential nature of cyber defense. However, its strength is not in static parsing, but in high-speed, sequential decision-making. Compelling quantitative evidence by Hussain et al. [2025] shows that RL-based systems can slash incident response times by 27.3% to 62% compared to traditional rule-based and supervised models.

For the task of parsing and representation, GNNs are the state-of-the-art. Source code is inherently a graph; GNNs are specifically designed to model the complex control flow, data flow, and dependency structures that define a program's behavior. The research strongly supports hybrid architectures that combine GNNs for representation and other models for decision-making.

To this end, we propose a hybrid Multi-Agent Reinforcement Learning (MARL) + Graph Neural Network (GNN) framework for real-time malware detection and adaptive response, augmented by an LLM-in-the-loop repair system. In our approach, GNNs analyze code repositories to reason about control flow, dependency structure, and potential infection propagation, providing a rich, high-fidelity state representation. This state is used by a team of MARL agents, which learn a coordinated policy to monitor and defend distributed code environments. When a threat is detected, a controller agent orchestrates the response: isolating compromised modules and invoking an LLM repair agent. This specialized agent, informed by the forensic data, synthesizes and tests patches in a sandbox environment before safe deployment.

This closed-loop architecture enables a continuous cycle of detection, containment, and autonomous remediation. It advances the field of cyber defense from static classification to an adaptive, interpretable, and self-reinforcing intelligence. Our framework thus moves toward a new paradigm of AI systems that can defend, heal, and evolve in real time against adversarial code.

2 Related Work

Recent advances demonstrate that reinforcement learning (RL) can outperform traditional static and supervised approaches in adaptive cybersecurity. McFadden et al. [2025] introduced DRMD , a PPO-based framework that unifies classification, active learning, and rejection to remain robust under concept drift. Similarly, Ouazzane et al. [2024], Dunsin et al. [2025] applied Q-learning to automate post-incident malware forensics, showing that RL agents can autonomously analyze memory dumps and behavioral traces, achieving rapid and adaptive threat identification. Mavroudis et al. [2025] formalized design guidelines for applying RL and multi-agent RL (MARL) in adversarial security settings, identifying challenges in scalability, coordination, and partial observability relevant to dynamic code analysis. Similarly, Tsingenopoulos et al. [2024] explored RL-based adversarial training within commercial antivirus pipelines, showing that problem-space perturbations can be leveraged to harden detectors without sacrificing functional correctness. Extending these ideas, Oprea et al. [2025] proposed PoolFlip, a MARL environment using Flip-PSRO to train defenders against evolving attacker populations, yielding generalizable and robust defensive policies through continuous self-play.

From the offensive perspective, Evans et al. [2018] showed that an RL agent can learn functionality-preserving transformations of Portable Executable files to evade static malware detectors, a capability later refined by Fang et al. [2020] with RLAttackNet, which improves detector robustness after adversarial retraining.Khan et al. [2025] further demonstrated that a D3QN agent can dynamically select minimal feature subsets for malware classification with near-perfect accuracy, significantly reducing computational overhead. Complementarily, Islam et al. [2024] combined PPO with large language models (LLMs) for automated vulnerability repair, integrating functional correctness and security as joint optimization objectives.

Our updated literature review has prompted us to refine our original hypothesis toward a more modular, multi-agent, and hybrid framework. Singh et al. [2024] introduced a hierarchical MARL model for cyber defense that decomposes complex network tasks into coordinated sub-agents, showing improved scalability and interpretability. This motivates extending our approach beyond a single RL agent toward multi-agent modularity, where agents specialize in distinct stages of malware investigation.

Reward function design is a well-recognized challenge in cybersecurity: O'Connell et al. [2025] empirically shows that sparse rewards foster robust RL defense policies in adversarial settings. In

Table 1: Comparative overview of cybersecurity paradigms and their trade-offs.

Methodology	**Speed & Scalability**	**Resource Cost**	**Efficacy vs. Zero-Day Threats**	**Resilience to Evasion**	**Primary Use Case**
Static Analysis	Very High	Low	Low	Low (vulnerable to packing/obfuscation)	First-line, large-scale code scanning; compliance
Dynamic Analysis	Low	High	High	Medium (vulnerable to sandbox detection)	Deep analysis of high-risk files; zero-day discovery
Supervised ML	High (Inference), High (Training)	High	Very Low	High (for known patterns)	Detecting variants of known malware families at scale
Reinforcement Learning	Medium (Inference), Very High (Training)	High	High	High (adaptive nature)	Automated incident response; dynamic threat hunting
MARL	High	High	Very High	Very High (coordinated adaptability)	Multi-agent defense coordination; red-blue training
Hybrid RL–GNN (Ours)	High	High	Very High	Very High (structural generalization)	Modular, interpretable post-incident analysis

our system, this translates to a multi-objective reward function with infrequent but significant feedback—penalizing missed threats heavily and discouraging indiscriminate quarantining, optimizing long-horizon defensive behavior.

Li et al. [2025] directly addresses drift resilience, demonstrating the effectiveness of domain-adaptive GNNs for maintaining robust malware detection amid changing code structures. Integrating such techniques into our GNN-based classifier further validates our approach to handling concept drift and data imbalance within evolving repositories. Similarly, Herath et al. [2025] introduce advances in explainability with CFGExplainer, demonstrating how interpretable control flow graph features allow GNN-based systems to highlight the specific code substructures responsible for detection decisions. This capability is critical for both trust and operational integration, as it enables reviewers to understand and critically assess automated quarantine and remediation actions.

Landolt et al. [2025] consolidates these findings by surveying MARL within cybersecurity, advocating for a decentralized, collaborative, and adaptive defense strategy. Their results support our phased approach: starting with a single RL agent proof-of-concept, scaling to a modular hierarchy where coordinated agents can partition, specialize, and interpret complex threats.

Synthesizing these insights, we revise our hypothesis: effective post-incident malware investigation should leverage a hybrid RL–GNN architecture within a modular multi-agent system, enabling dynamic, explainable, and adaptive threat analysis across evolving code repositories. Continuous adversarial training ensures robustness against emerging and adaptive threats, with architecture choices directly validated and motivated by the recent literature. We summarize the trade-offs of these paradigms in Table 1.

3 Proposed Experimental Methodology

3.1 Phase 1: Experimental Architecture & Setup

Our experiment begins by building a "Digital Twin" CI/CD pipeline. This is a closed-loop, simulated environment, containerized with open-source tools like Docker. It acts as an interactive "Gym" where AI agents can learn and operate, mirroring the full lifecycle of code changes in a real software repository. Within this environment, we deploy three core AI components.

Component 1: The GNN State Encoder (The "Eyes")

This component is responsible for understanding incoming code changes. When a `git push` event occurs, the Encoder parses the code *diff* and transforms it into its fundamental graph structures: Abstract Syntax Trees (ASTs) to represent grammatical structure, Control Flow Graphs (CFGs) for execution paths, and Data Flow Graphs (DFGs) to track data dependencies. These graphs are then fed into a pre-trained GNN model which outputs a fixed-length vector embedding. This embedding serves as the primary State Representation (S) for the MARL agents.

Component 2: The MARL Defense Swarm (The "Brain")

This component orchestrates the defense as a hierarchical Multi-Agent Reinforcement Learning (MARL) system with two agents. The Observer agent receives the GNN state vector (S) for each code change and performs rapid threat detection, taking one of three actions: `Malicious`, `Suspicious`, or `Benign`. It learns via a dense reward system, with heavy penalties for false negatives and smaller penalties for false positives. Conversely, the Controller agent activates only if the Observer flags a commit. It uses the GNN state (S) and the Observer's flag to choose higher-level orchestration actions: `Isolate_Module`, `Block_Commit`, or `Invoke_Repair`. The Controller is trained with a sparse, high-impact reward delivered only at the end of successful remediation, encouraging it to learn robust, long-term strategies.

Component 3: The LLM Repair Agent (The "Hands")

This is a non-learning tool invoked when the Controller decides `Invoke_Repair`. Its process is sequential. First, it performs Context Assembly by gathering the vulnerable code snippet, GNN analysis, and forensic data (e.g., malicious API traces from a sandbox execution) to create a detailed LLM prompt. Second, the LLM generates a candidate patch using this context. Third, this patch

undergoes Sandbox Validation in a new isolated environment. This validation runs two suites: (1) functional unit tests to ensure existing functionality is preserved and (2) security tests that re-run the original attack payload to verify effectiveness. Finally, the binary validation result (`Pass/Fail`) is reported back to the MARL Controller as its reward, completing the autonomous detection-to-remediation loop.

3.2 Phase 2: Baselines, Datasets, and Procedure

We benchmark against three baselines: (1) a top-tier Static Application Security Testing tool, (2) a standalone GNN classifier, representing simple ML detection, and (3) our GNN+MARL framework without the LLM agent. Experiments use a large dataset of benign code commits and two-part malicious datasets: known Common Vulnerabilities and Exposures (e.g., the EMBER dataset [Anderson and Roth, 2018]), and novel adversarial payloads to simulate zero-day threats. We will run 10,000 CI/CD simulations with a 90/10 benign-to-malicious ratio, a standard practice for RL training under imbalanced cybersecurity data.

3.3 Phase 3: Metrics and Validation

We evaluate detection efficacy using Precision, Recall, F1-Score, and False Positive Rate (FPR). Response speed is measured by Mean Time to Remediation (MTTR), from commit to fully patched validation, while autonomous robustness is assessed via Zero-Day Detection Rate and Autonomous Repair Rate. Our hypothesis is supported if the framework shows statistically significant reductions in MTTR, superior F1 and Zero-Day Detection scores, and a non-zero Autonomous Repair Rate, all while maintaining an operationally viable FPR.

4 Limitations of the Proposed Method

Despite its rigor, this design has limitations. The Digital Twin cannot fully capture the stochastic complexity of real-world, human-driven software development. The 10% malicious event rate, necessary for RL training, does not reflect true-world imbalances, potentially inflating false positives. Framework robustness depends on the quality of MARL reward functions; sparse and hierarchical rewards introduce challenges such as reward hacking and convergence instability. Computational demands are significant, with GNN embedding, MARL policy execution, and LLM-driven patch generation creating high latency and interpretability issues. Finally, the autonomous repair loop assumes comprehensive test suites; insufficient coverage may allow LLM-generated patches to pass validation while introducing new functional or security vulnerabilities.

References

A. Hussain, L. Wang, S. Ahmed, and M. Yang. Optimizing cybersecurity incident response via adaptive reinforcement learning. *Journal of Advanced Engineering and Technology*, 2(1):45–59, 2025. doi: 10.62177/jaet.v2i1.212. URL https://doi.org/10.62177/jaet.v2i1.212.

Shae McFadden, Myles Foley, Mario D'Onghia, Chris Hicks, Vasilios Mavroudis, Nicola Paoletti, and Fabio Pierazzi. Drmd: Deep reinforcement learning for malware detection under concept drift, 2025. URL https://arxiv.org/abs/2508.18839.

Karim Ouazzane, Mohamed Chahine Ghanem, Dipo Dunsin, and Vassil Vassilev. Reinforcement learning for an efficient and effective malware investigation during cyber incident response, 2024. URL https://arxiv.org/abs/2408.01999.

Dipo Dunsin, Mohamed Chahine Ghanem, Karim Ouazzane, and Vassil Vassilev. A novel reinforcement learning model for post-incident malware investigations, 2025. URL https://arxiv.org/abs/2410.15028.

Vasilios Mavroudis, Gregory Palmer, Sara Farmer, Kez Smithson Whitehead, David Foster, Adam Price, Ian Miles, Alberto Caron, and Stephen Pasteris. Guidelines for applying rl and marl in cybersecurity applications, 2025. URL https://arxiv.org/abs/2503.04262.

Ilias Tsingenopoulos, Jacopo Cortellazzi, Branislav Bošanský, Simone Aonzo, Davy Preuveneers, Wouter Joosen, Fabio Pierazzi, and Lorenzo Cavallaro. How to train your antivirus: Rl-based hardening through the problem space. In *Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses*, RAID '24, page 130–146, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400709593. doi: 10.1145/3678890.3678912. URL https://doi.org/10.1145/3678890.3678912.

Alina Oprea, Peter Chin, Simona Boboila, Xavier Cadet, and Sie Hendrata Dharmawan. Poolflip: A multi-agent reinforcement learning security environment for cyber defense, 2025. URL https://arxiv.org/abs/2508.19488.

David Evans, Bobby Filar, Anant Kharkar, Hyrum S. Anderson, and Phil Roth. Learning to evade static pe machine learning malware models via reinforcement learning, 2018. URL https://arxiv.org/abs/1801.08917.

Yong Fang, Yuetian Zeng, Beibei Li, Liang Liu, and Lei Zhang. Deepdetectnet vs rlattacknet: An adversarial method to improve deep learning-based static malware detection model. *PLOS ONE*, 15:e0231626, 2020. doi: 10.1371/journal.pone.0231626. URL https://doi.org/10.1371/journal.pone.0231626.

Naseem Khan, Aref Y. Al-Tamimi, Amine Bermak, and Issa M. Khalil. Adaptive malware detection using sequential feature selection: A dueling double deep q-network (d3qn) framework for intelligent classification, 2025. URL https://arxiv.org/abs/2507.04372.

Nafis Tanveer Islam, Mohammad Bahrami Karkevandi, and Peyman Najafirad. Code security vulnerability repair using reinforcement learning with large language models, 2024. URL https://arxiv.org/abs/2401.07031.

Aditya Vikram Singh, Ethan Rathbun, Emma Graham, Lisa Oakley, Simona Boboila, Peter Chin, and Alina Oprea. Hierarchical multi-agent reinforcement learning for cyber network defense. *arXiv preprint arXiv:2410.17351*, 2024. URL https://arxiv.org/pdf/2410.17351.

S. O'Connell, S. M. Smith, and T. Jones. Less is more? rewards in rl for cyber defence. *arXiv preprint arXiv:2503.03245*, 2025. URL https://arxiv.org/pdf/2503.03245.

Adrian Shuai Li, Arun Iyengar, Ashish Kundu, and Elisa Bertino. Revisiting concept drift in windows malware detection: Adaptation to real drifted malware with minimal samples. *arXiv preprint arXiv:2502.10556*, 2025. URL https://arxiv.org/pdf/2502.10556.

Jerome Dinal Herath, Priti Prabhakar Wakodikar, Ping Yang, and Guanhua Yan. Cfgexplainer: Explaining graph neural network-based malware classification from control flow graphs. *arXiv preprint arXiv:2504.16316*, 2025. URL https://arxiv.org/pdf/2504.16316.

Christoph R. Landolt, Christoph Würsch, Roland Meier, Alain Mermoud, and Julian Jang-Jaccard. Multi-agent reinforcement learning in cybersecurity: From fundamentals to applications. *arXiv preprint arXiv:2505.19837*, 2025. URL https://arxiv.org/abs/2505.19837.

Hyrum S. Anderson and Phil Roth. Ember: An open dataset for training static pe malware machine learning models, 2018. URL https://arxiv.org/abs/1804.04637.