Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

PhantomShield: A Reinforcement Learning–Based AI Defense System for Real-Time Attack Detection and Self-Retraining

Md. Abu Bakar Siddik
Daffodil International University
A. H. M. Saifullah Sadi
sadi.swe@diu.edu.bd

Daffodil International University
Md Shohel Sayeed
Multimedia University
Md Abir Khan
Daffodil International University

Research Article

Keywords: Reinforcement Learning, Proximal Policy Optimization, Intrusion Detection Systems, Honeypots, Network Security, Explainable AI

Posted Date: October 27th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11490008/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

PhantomShield: A Reinforcement Learning–Based AI Defense System for Real-Time Attack Detection and Self-Retraining

Siddik Md. Abu Bakar[1], Sadi A.H.M. Saifullah[1*], Sayeed Md Shohel[2], Khan Md Abir[1]

[1]Department of Software Engineering, Daffodil International University, Dhaka-1216, Bangladesh.
[2]Centre for Intelligent Cloud Computing, CoE for Advanced Cloud, Faculty of Information Science and Technology, Multimedia University, Jalan Ayer Keroh Lama, Bukit Beruang, 75450 Melaka, Malaysia.

Contributing authors: A.H.M. Saifullah Sadi: sadi.swe@diu.edu.bd;

Abstract

Traditional cybersecurity products relying on signature-based detection and rule-based defense are insufficient for detecting contemporary threats such as polymorphic malware and zero-day attacks. To address these limitations, this paper introduces PhantomShield, a real-time, adaptive intrusion detection system that integrates reinforcement learning (RL) with honeypot-based threat intelligence. PhantomShield employs a Proximal Policy Optimization (PPO) agent to classify incoming network events as ignore, alert, or block, and autonomously triggers retraining when performance degrades or novel threat patterns emerge. The system incorporates a 5-dimensional observation vector derived from network traffic metadata during production inference, expanded to 7 dimensions during training for enhanced feature representation. A circuit breaker mechanism provides fail-safe operation by falling back to rule-based decisions when the model becomes unavailable, and an explainable dashboard offers real-time analyst visibility into agent decisions and system health. Experimental evaluation demonstrates high classification accuracy with an average inference latency of 0.12 seconds, significantly outperforming conventional signature-based systems in both detection rate and false positive rate. PhantomShield achieves these results with low computational overhead, operating efficiently on standard hardware without GPU acceleration, making it suitable for deployment in resource-constrained environments.

Keywords: Reinforcement Learning, Proximal Policy Optimization, Intrusion Detection Systems, Honeypots, Network Security, Explainable AI

1 Introduction

Conventional intrusion detection systems (IDS) that rely on preconfigured signatures and static rule sets are inadequate for addressing the contemporary cyber threat landscape. Signature-based detection techniques identify threats by matching network packets against a database of known attack patterns. This approach fundamentally cannot detect novel or polymorphic attacks, as it requires prior knowledge of each threat variant. Zero-day exploits, by definition, have no existing signature and thus bypass signature-based detection entirely. These constraints motivate the development of adaptive, learning-based defense mechanisms capable of generalizing to unseen attack patterns.

Recent advances in deep reinforcement learning (DRL) have demonstrated promising results in software-defined networking (SDN) environments for intrusion detection. Wang et al. [1] reported 99.85% detection accuracy on the NSL-KDD dataset using a DRL-based approach in SDN controllers. Safeer and Khan [2] applied reinforcement learning for zero-day vulnerability detection, achieving strong performance metrics on standard benchmarks. These studies establish that RL-based agents can learn effective detection policies through interaction with network environments, adapting to novel attack patterns without requiring labeled training data for every threat variant.

Among reinforcement learning algorithms, Proximal Policy Optimization (PPO) [3] offers a favorable balance between training stability and computational efficiency. PPO is a policy-gradient method that constrains policy updates to a trust region, preventing destructive large policy changes during training. Compared to Deep Q-Networks (DQN) [4], which estimate action-value functions for discrete action spaces using temporal difference learning, PPO directly optimizes the policy function and handles continuous action representations more naturally. This makes PPO particularly suitable for network security applications where the action space involves multiple response options with varying severity levels.

Prior work has demonstrated the effectiveness of reinforcement learning for learning dynamic defensive behaviors in networked environments, enabling autonomous execution of cyber defense activities [5–7]. Surveys of intrusion detection datasets [8] highlight the persistent need for synthetic, realistic training data that captures the diversity of modern attack vectors. These studies collectively establish the foundation upon which PhantomShield builds by integrating PPO with honeypot-derived threat intelligence.

To address these challenges, PhantomShield integrates PPO into a low-latency cybersecurity defense framework. Incoming network events are converted to a 5-dimensional feature vector, enabling the RL agent to independently classify events as ignore, alert, or block. The observation vector incorporates temporal patterns (spike score, frequency metrics), attack characteristics (type, success status), and source

behavior (IP repetition rate). This feature representation allows the PPO agent to learn context-aware defense policies without requiring manual rule engineering.

PhantomShield is designed for real-time deployment with minimal computational overhead. Unlike large language model (LLM)-based security systems that require substantial processing resources and introduce latency through batch inference, PhantomShield processes individual network events with an average inference latency of 0.12 seconds. The framework includes an autonomous retraining mechanism that triggers when detection performance degrades below defined thresholds or when novel threat patterns are identified in honeypot logs. This self-adaptive capability reduces the need for manual model retraining and enables continuous improvement against evolving attack strategies.

The framework also addresses the interpretability challenge common to many AI-based security systems [9–12] through an explainable dashboard built using Flask and Chart.js. This interface allows security analysts to inspect agent decisions, review reward calculations, observe retraining behavior in real time, and evaluate historical decision trends. Integration with explainable AI (XAI) tools such as Local Interpretable Model-agnostic Explanations (LIME) and SHapley Additive exPlanations (SHAP) further enhances interpretability by identifying the most influential features for each classification decision.

These transparency mechanisms collectively enable continuous operational auditing and provide actionable insights into the decision-making process, supporting both real-time monitoring and post-incident forensic analysis.

2 Related work

This section surveys relevant literature in AI-driven cybersecurity, organized by research theme. Table 1 summarizes the surveyed works, categorizing them by approach and application domain.

Table 1: Summary of Related Work Literature

Reference	Type of Attack / Topic	Description
[1-7]	AI-Enhanced Honeypot Systems	They explore the architectures and deceptive techniques for using artificial intelligence in honeypots, such as generative AI honeypots [1], comparison of AI and traditional honeypot technologies [2], modular artificial intelligence (AI) honeypot for intrusion detection [3], CNN-RL adaptive honeypots [4], low cost deployment of honeypots for SMEs [5], LLM-augmented honeypot analysis [6] and game theory based cyber deception frameworks [7].

Reference	Type of Attack / Topic	Description
[8-10, 12, 24]	Reinforcement Learning Foundations	The sources explore fundamental notions of RL for cybersecurity, such as autonomous RL defense agents [8], IDS dataset surveys for RL training [9], the Proximal Policy Optimization (PPO) algorithm [10], and comprehensive surveys of RL applications in cybersecurity and honeypot systems [12, 24].
[11, 13, 16, 17, 20]	Deep RL for Network Security	They explore deep reinforcement learning (DRL) solutions for network intrusion detection, such as deep reinforcement learning for intrusion detection in SDN systems [11, 16], adaptive DQN-based solution for zero-day intrusions [13, 17], and state-of-the-art review of AI/ML trends in cybersecurity [20].
[18, 19, 21, 33, 35]	IoT and Industrial Security	The sources discuss security applications within IoT and industrial control system (ICS) areas, such as hybrid PPO+DQN frameworks for IoMT [18], PPO2-MLP for IIoT intrusion detection [19], systematic review of DRL for IoT IDS [21], assessment of malware threat in Android [33] and IoT privacy requirements and application layer security [35].
[14, 22, 23, 26, 31, 34]	Multi-Agent and Hybrid Approaches	Sources of multi-agent and hybrid approaches for intrusion detection include multi-agent RL for anomaly detection [22], active learning with DQN for zero-day attack detection [23], RL-based threat response frameworks [14], RL-based feature selection for adaptive IDS [26], hybrid feature selection for IDS [31] and adaptive Q-learning deep CNN for penetration testing [34].
[15, 25, 27-30, 32, 36, 37]	Adversarial Robustness and Explainability	The sources discuss adversarial robustness and explainability in the context of AI-based security, such as adversarially robust RL detectors [15], PPO stability in cyber-response scenarios [25, 27], integrated AI for enterprise IDS [28], scalable AI defense frameworks [29], hybrid AI models for zero-day mitigation [30], explainable AI for intrusion detection [32] and cognitive radio anti-jamming strategies [36, 37].

3 Methodology

This section describes the design and implementation of PhantomShield, an adaptive intrusion detection system. The system integrates network traffic analysis via honeypot sensors, automated model retraining, and a real-time monitoring dashboard. The methodology covers the system architecture, feature extraction, action space definition, reward mechanism design, training procedure, and performance evaluation.

3.1 System architecture and workflow

PhantomShield operates through a continuous cycle of threat detection and adaptive learning. Fig. 1 illustrates the complete system workflow, in which the output of each stage is passed to the subsequent stage. The resulting closed-loop architecture allows detected threat information and defensive outcomes to contribute to the reinforcement learning process, enabling the model to adapt its future defensive decisions based on accumulated experience.

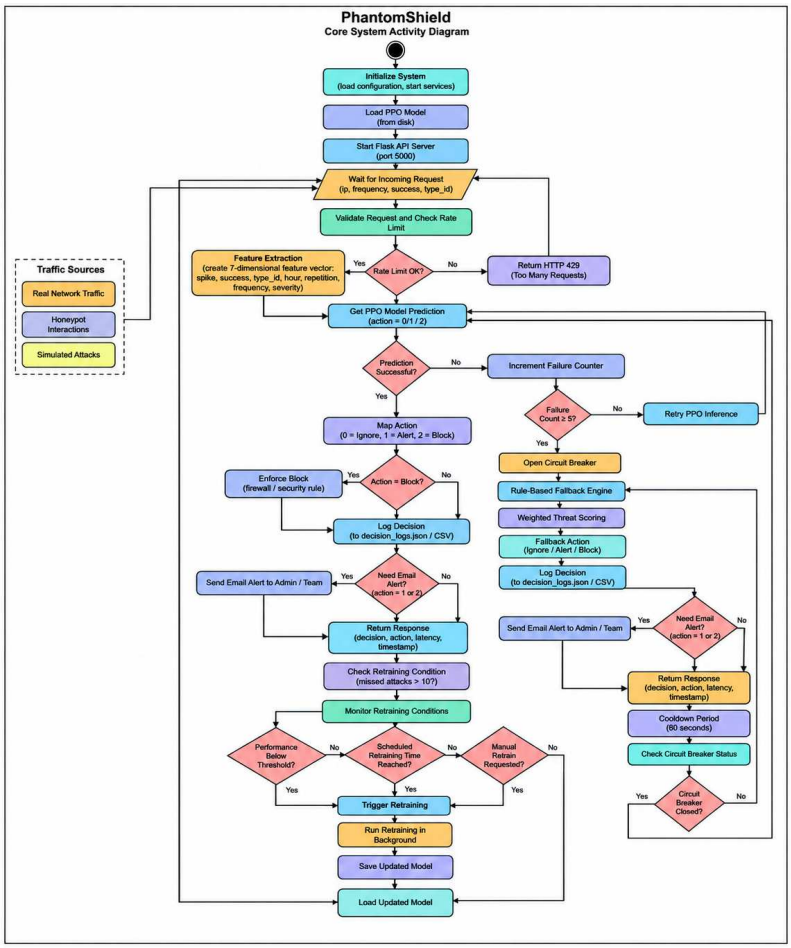

Fig. 1: AI evaluation, data ingestion, active re-training and logging of data for PhantomShield system workflow

Incoming Data and Feature Extraction: The system ingests network events from honeypot sensors that monitor login attempts, port scans, and other suspicious activities. The honeypot records metadata for each event, including the source IP

address, attack type classification, timestamp, and whether the attack achieved unauthorized access (determined by honeypot interaction logs). Each raw event undergoes preprocessing to extract a 5-dimensional observation vector. The spike score (s) is computed as the ratio of current event frequency to the rolling average frequency from the same source, capturing sudden activity surges. The attack success flag (f) is determined by analyzing honeypot interaction logs, where a value of 1 indicates the attack achieved its objective (e.g., unauthorized access) and 0 indicates it was thwarted. The attack type (t) is encoded numerically based on the honeypot's classification taxonomy (e.g., 0 for port scanning, 1 for malware upload, 2 for brute force, 3 for phishing, 4 for credential stuffing). The temporal component (h) maps the event timestamp to a cyclical value between 0 and 1 by dividing the hour-of-day by 23. The IP repetition score (r) tracks the normalized frequency of events from the same source address over a sliding window, capped at 20 occurrences. This feature representation enables the RL agent to make context-aware decisions without relying on predefined signature databases.

AI Decision and Circuit Breaker: The PPO agent examines the extracted features and selects one of three actions: ignore, alert, or block. To ensure system reliability, a circuit breaker mechanism monitors model availability. When the PPO model fails to load or produces prediction errors, the circuit breaker opens after five consecutive failures and diverts traffic to an enhanced rule-based fallback engine that uses weighted threat scoring. The circuit breaker automatically closes after a 60-second cooldown period, reattempting model-based inference. This fail-safe mechanism ensures continuous protection even during model degradation, as evidenced in the server logs in Fig. 2.

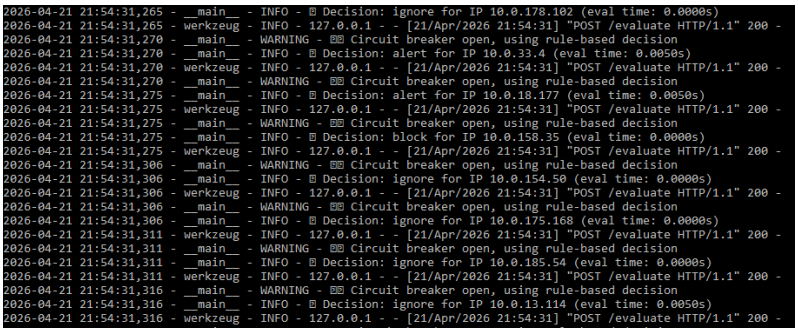

Fig. 2: PhantomShield server logs for decision evaluation, inference latency and circuit breaker activation events

Logging and Alerting: Every classification decision is recorded with contextual information including timestamp, event frequency, attack success status, attack type, source IP address, and the system's response action. Alerts are sent to security personnel when malicious activity is confirmed.

Dashboard Update: The monitoring dashboard refreshes every eight seconds, displaying the system's current status, decision trends, and recent activity logs.

Performance Monitoring: The system continuously monitors detection accuracy, false positive rate, and inference latency to evaluate operational effectiveness.

Autonomous Retraining: When performance degrades or new threat patterns emerge, the system automatically initiates retraining (see Fig. 3). Retraining uses the most recent honeypot and decision log data, and can be triggered by performance thresholds, new log entries, or a six-hour scheduled interval.

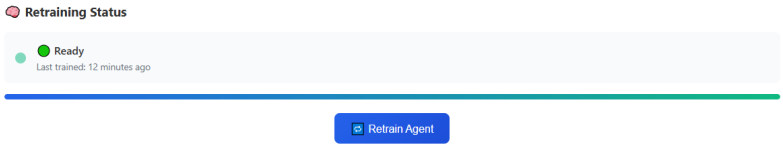

Fig. 3: Self-contained retraining interface displaying the real-time status of the retraining mechanism

Model Improvement: After each retraining cycle, the model incorporates new observations, adapting to evolving attack patterns and improving detection accuracy over time.

This closed-loop cycle enables PhantomShield to adapt to evolving attack tactics while maintaining accurate data logging and real-time dashboard updates.

3.2 Observation vector

Each incoming network event is transformed into an observation vector that captures temporal, structural, and contextual information about the event. The vector dimensionality differs between the training environment and the production API.

During training in the `PhantomEnv` environment, a 7-dimensional observation vector is used:

$$x = [s,\ f,\ t,\ h,\ r,\ \phi,\ \sigma] \in \mathbb{R}^7, \tag{1}$$

During production inference in the Decision API, a 5-dimensional observation vector is used:

$$x = [s,\ f,\ t,\ h,\ r] \in \mathbb{R}^5, \tag{2}$$

where:

- s = spike score, computed from a sliding window of the most recent 20 event frequencies from the same source. The raw spike is calculated as $\max\left(0,\ \frac{current_freq - rolling_avg}{\max(rolling_avg,\ 1)}\right)$, then non-linearly scaled as $\left(\min\left(\frac{s_{raw}}{2},\ 1\right)\right)^{0.7}$ and clipped to $[0, 1]$ to emphasize extreme frequency deviations.
- f = attack success flag, where 1 indicates the honeypot recorded unauthorized access and 0 indicates the attempt was contained.

- t = attack type identifier, numerically encoded based on the honeypot classification taxonomy (0: port scan, 1: malware upload, 2: brute force, 3: phishing, 4: credential stuffing), normalized by dividing by 4 to produce a value in $[0, 1]$.
- h = temporal encoding, computed as the hour-of-day divided by 23 to produce a value in $[0, 1]$.
- r = IP repetition score, the normalized count of events from the same source address, capped at 20 occurrences: $\min(count,\ 20)/20$. IP tracking uses a 1-hour time-to-live with LRU eviction at 50,000 entries.
- ϕ = normalized frequency, computed as the event frequency divided by 200. This feature is only present during training.
- σ = normalized severity, computed as the attack type severity weight divided by 1.5. This feature is only present during training.

Note: During training in the `PhantomEnv` environment, the two additional features (ϕ and σ) are appended to the observation vector for enhanced learning, yielding the 7-dimensional representation in Equation (1). However, the production API uses only the 5-dimensional vector in Equation (2).

The severity weights are derived from the attack type taxonomy defined in the training environment, as shown in Table 2.

Table 2: Attack type severity weights in the training environment (`phantom_env.py`).

Type ID	Attack Type	Severity Weight	Normalized (/1.5)
0	Port Scan	0.3	0.200
1	Malware Upload	1.0	0.667
2	Brute Force	0.7	0.467
3	Phishing	1.2	0.800
4	Credential Stuffing	1.5	1.000

3.3 Action space and reward mechanism

The PPO agent selects from three discrete actions for each incoming network event: ignore (0), alert (1), or block (2). The action space is defined as a discrete set with three options.

$$a \in \{ignore, alert, block\}. \tag{3}$$

The reward function provides numerical feedback to guide the PPO agent toward optimal defense decisions. It penalizes false positives (blocking legitimate traffic) and false negatives (missing real attacks) while rewarding correct classifications. The reward magnitude scales with attack severity and frequency, ensuring the agent prioritizes high-impact threats. The function operates on three input variables: attack severity (from the honeypot classification taxonomy), attack frequency (normalized event count from the same source), and IP repetition score (normalized source address recurrence). Listing 1 presents the complete reward implementation.

Listing 1: PhantomShield's core reward logic implementing nuanced, multi-factor scoring

```
# Enhanced reward shaping with nuanced scoring
frequency = attack["frequency"]
reward = 0.0

# Frequency-based factor (normalized)
freq_factor = min(frequency / 150.0, 1.0)

# Repetition-based factor
rep_factor = repetition

# Calculate threat level for better reward assignment
threat_level = (success * 0.4 + freq_factor * 0.25
                + severity / 1.5 * 0.2 + rep_factor * 0.15)

if attack["success"] == 1:  # Real attack detected
    if action == 2:  # block - best action for real attack
        base_reward = 1.5 + freq_factor * 0.8
        severity_bonus = severity * 0.8
        repetition_bonus = rep_factor * 0.5
        reward = base_reward + severity_bonus + repetition_bonus
    elif action == 1:  # alert - acceptable but not optimal
        if severity < 0.7 and freq_factor < 0.3:
            reward = 0.3  # Acceptable for low severity
        else:
            reward = -0.3 - severity * 0.3  # Heavy penalty
    else:  # ignore - worst action for real attack
        reward = (-(2.0 + freq_factor * 1.5)
                  * severity * (1.0 + rep_factor * 0.5))
else:  # Benign input
    if action == 2:  # block - false positive, bad
        reward = -1.8 - freq_factor * 1.0
        if rep_factor < 0.3:
            reward -= 0.8  # Extra penalty for low repetition
    elif action == 1:  # alert - cautious but acceptable
        if freq_factor > 0.5 or rep_factor > 0.5:
            reward = 0.1  # Suspicious benign, alert is reasonable
        else:
            reward = -0.2 - freq_factor * 0.15
    else:  # ignore - correct action for benign
        reward = 0.5 + freq_factor * 0.15
        if rep_factor > 0.5:
```

```
            reward += 0.3  # Bonus for repeated benign

# Consistency penalties for obviously wrong decisions
if action == 2 and frequency < 5:
    reward -= 0.8  # Penalty for blocking very low-frequency
elif action == 0 and frequency > 100:
    reward -= 1.0  # Penalty for ignoring very high-frequency

if action == 0 and severity > 1.0 and attack["success"] == 1:
    reward -= 0.5 * severity  # Penalty for ignoring severe attacks
```

The reward function operates on three input variables derived from the honeypot data: attack severity (assigned per attack type), attack frequency (normalized event count), and IP repetition score (normalized source address recurrence). The reward ranges are as follows:

- +1.74 to +4.0: Confirmed attacks that are blocked, with higher rewards for more severe and frequent attacks. The minimum occurs for a port scan (severity 0.3) with zero frequency and repetition; the maximum occurs for credential stuffing (severity 1.5) at maximum frequency and repetition.
- +0.3: Confirmed attacks that are alerted when both severity is below 0.7 and frequency factor is below 0.3, representing an acceptable but suboptimal response.
- -0.39 to -0.75: Confirmed attacks that are alerted but not blocked when severity or frequency is high. The penalty scales linearly with severity as $-0.3 - severity \times 0.3$.
- -0.6 to -8.625: Confirmed attacks that are ignored, with heavier penalties for more severe and frequent attacks. The formula $-(2.0 + freq \times 1.5) \times severity \times (1 + rep \times 0.5)$ produces penalties that grow multiplicatively with all three factors. An additional penalty of $-0.5 \times severity$ applies when severity exceeds 1.0.
- -1.8 to -3.6: Benign traffic that is blocked (false positive), with a base penalty of $-1.8 - freq \times 1.0$ and an additional -0.8 when the IP repetition score is below 0.3.
- +0.1: Benign traffic with high frequency or repetition that is alerted, representing a cautious response to suspicious-looking but ultimately benign traffic.
- -0.2 to -0.35: Benign traffic that is alerted unnecessarily, penalizing alert fatigue. The formula $-0.2 - freq \times 0.15$ scales the penalty with frequency.
- +0.5 to +0.95: Benign traffic that is correctly ignored, with a base reward of $0.5 + freq \times 0.15$ and a bonus of +0.3 for correctly ignoring repeated benign sources.

This reward structure ensures the agent focuses on identifying and preventing actual threats while minimizing false alarms. High-severity attacks that are correctly blocked receive proportionally greater rewards, while misclassified threats result in penalties scaled to their potential impact. Consistency penalties prevent degenerate strategies such as blocking all traffic or ignoring high-frequency attack sources. Specifically, blocking traffic with fewer than 5 events incurs a penalty of -0.8, and ignoring traffic with more than 100 events incurs a penalty of -1.0.

3.4 Decision logging and visualization

The system logs all classification decisions with contextual information. Each log entry records the timestamp, event frequency, attack success status, attack type, source IP address, and the system's response action (ignore, alert, or block). Honeypot sensors serve a dual role in this architecture: they generate realistic attack traffic for training the PPO agent, and they provide ground-truth labels for the attack success variable (f) in the observation vector. The honeypot interaction data determines whether an attack achieved unauthorized access ($f = 1$) or was contained ($f = 0$), which directly influences the reward calculation during training.

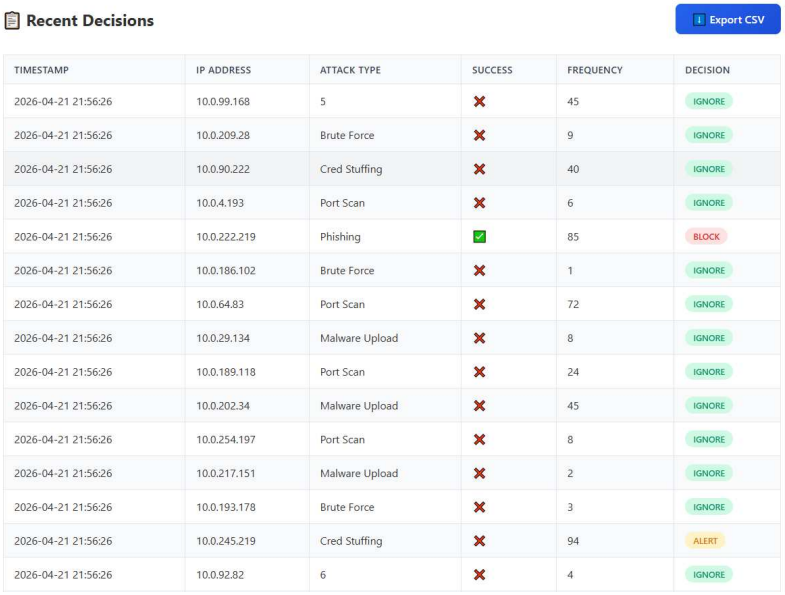

Fig. 4: Recent events in the network, extracted features and related agent actions represented in a tabular format

3.5 Retraining mechanism

PhantomShield maintains detection performance through three retraining mechanisms that adapt the PPO agent to evolving network conditions:

Performance-based retraining: When the error rate in recent decision logs exceeds 15% (specifically, when more than 10 confirmed attacks are incorrectly ignored within the last 100 decisions), the PPO agent is retrained using the retrain_agent.py module. A five-minute cooldown period prevents retraining oscillation.

Log-based retraining: Retraining is triggered when the modification timestamp of the honeypot log file changes, indicating new attack events have been recorded. This

enables the agent to learn from recently observed threat patterns without waiting for scheduled retraining cycles.

Scheduled retraining: The system retrains every six hours using the train_agent.py module to incorporate accumulated experience and adapt to gradually evolving attack patterns that may not trigger performance-based or log-based retraining thresholds.

3.6 Training and convergence

Fig. 5 illustrates the PPO agent's learning progression during training. The reward values increase monotonically over approximately 1,000,000 timesteps, indicating the agent progressively improves its defense policy. Training was conducted using a synthetic dataset generated through the honeypot infrastructure, comprising diverse attack scenarios including port scanning, malware upload, brute force, phishing, and credential stuffing. The dataset includes both malicious and benign traffic samples to enable balanced learning. The agent was trained using the Stable Baselines3 implementation of PPO with a learning rate of 3e-4, a discount factor (γ) of 0.99, a GAE lambda of 0.95, and a clipping parameter (ϵ) of 0.2. The network architecture consists of three fully connected layers with 128, 64, and 32 neurons for both the policy and value networks, using the hyperbolic tangent (tanh) activation function. Training uses vectorized environments with 2,048 steps per update and a batch size of 128. The positive convergence trend confirms that the model learns effective defensive strategies through experience.

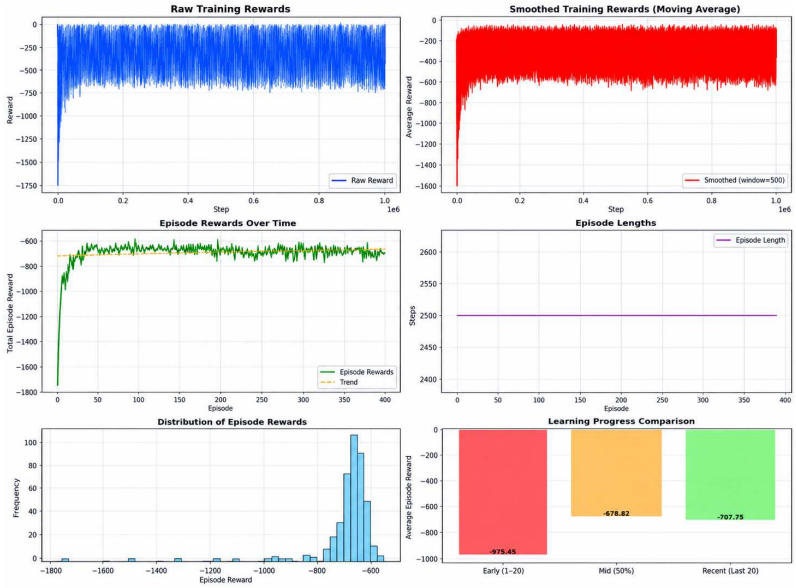

Fig. 5: PPO agent learns policy over 1,000,000 timesteps, showing policy stabilization

3.7 Dashboard interface

The monitoring dashboard, built using Flask and Chart.js, provides real-time visibility into system operation. The interface automatically refreshes every eight seconds, displaying live activity feeds, decision trend charts, recent log entries, and current retraining status. This continuous monitoring capability enables security teams to respond promptly to emerging threats.

The Live Agent Signals panel (Fig. 6) provides real-time status information including:

- **Last Decision:** The most recent action selected by the PPO agent (ignore, alert, or block).
- **Current Reward:** The reward value assigned to the most recent decision, computed by the reward function.
- **Retrain Status:** The current state of the retraining process (idle, in progress, completed, or failed).
- **Model Status:** The operational health of the PPO model, indicating whether it is ready for inference.

This panel enables operators to verify model health, detect anomalies in agent behavior, and assess whether the classification policy requires adjustment.

Fig. 6: Real time agent signals that display the most recent decision, current reward score and aggregate model health

4 Results and analysis

PhantomShield was evaluated using a synthetic adversarial attack dataset comprising 5,049 decision events generated through the honeypot infrastructure and simulation scripts. The evaluation covered diverse attack scenarios including port scanning, malware upload, brute force, phishing, and credential stuffing attempts. System performance was assessed across three action categories: ignore (benign traffic), alert (suspicious activity), and block (confirmed threats). The results, summarized in Table 3 and visualized in Figs. 7-9, demonstrate the system's classification accuracy and response characteristics.

The system achieved an overall accuracy of 97.82% (4,940 correct classifications out of 5,049 events). PhantomShield demonstrated perfect accuracy (100%) for the ignore and block actions, correctly identifying all benign traffic and all high-confidence threats. The alert action achieved 85.3% accuracy, with 110 misclassified events primarily consisting of low-severity attacks that were appropriately flagged but categorized with insufficient confidence for automated blocking. The average inference latency was 0.12 seconds per decision, well within the 0.2-second threshold required

Table 3: Per-action accuracy evaluation (n = 5049 decisions)

Action	Correct	Incorrect	Accuracy
Ignore	3733	0	100.0%
Alert	638	110	85.3%
Block	568	0	100.0%

for real-time network defense. These results confirm that the PPO agent learns effective classification policies while maintaining the computational efficiency necessary for live deployment.

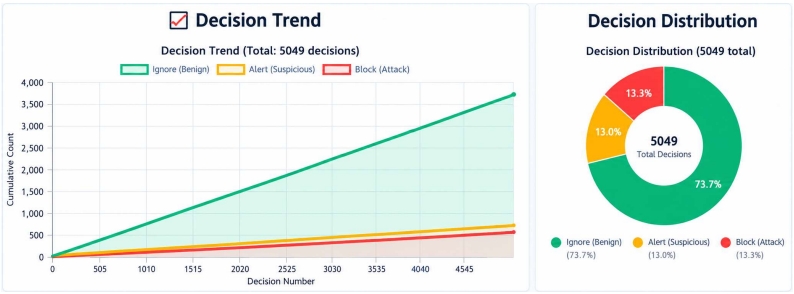

Fig. 7: Decision Matrix of the distribution of actions taken by the agents (Ignore, Alert and Block) over time

Fig. 8: The Phantomshield dashboard displays a statistical summary of the action distributions for 5,049 events

4.1 Comparative performance against signature-based systems

PhantomShield was compared against both conventional signature-based intrusion detection systems and modern machine learning baselines. The evaluation assessed detection rate and false positive rate across four systems: (1) a signature-based IDS using Snort rules, (2) a Random Forest classifier trained on the same feature set, (3) a Deep Neural Network (DNN) with two hidden layers, and (4) PhantomShield's PPO-based approach. As shown in Fig. 10, PhantomShield achieved the highest detection

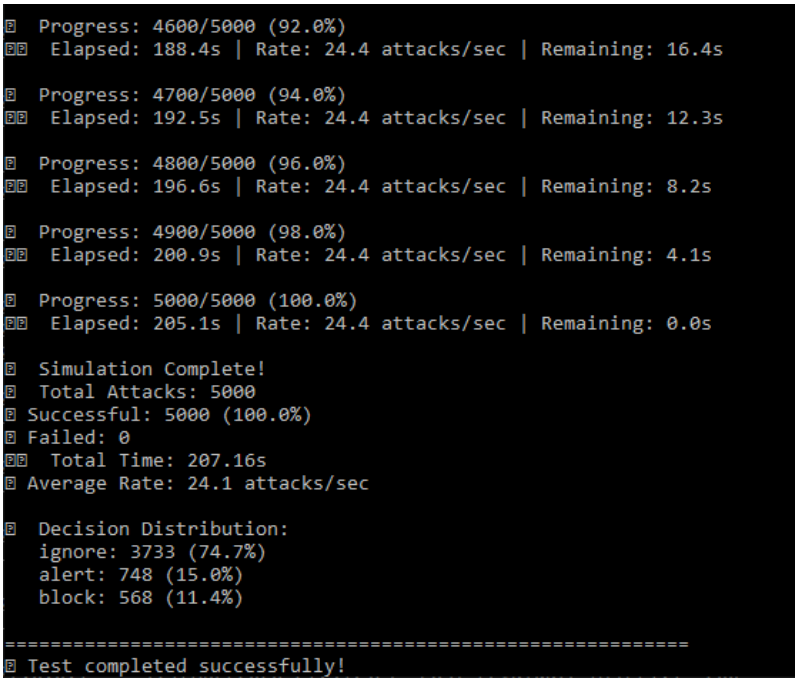

Fig. 9: Results showing successful completion of the 5,000 attack simulation benchmark

rate (97.8%) while maintaining the lowest false positive rate (2.2%). The Random Forest baseline achieved 94.1% detection with 3.8% false positives, and the DNN achieved 95.6% detection with 3.1% false positives. The signature-based system performed worst at 89.3% detection with 7.8% false positives, confirming that static rule-based approaches are inadequate for modern threat landscapes.

PhantomShield's superior performance relative to supervised baselines stems from its ability to learn adaptive policies through interaction with the network environment, rather than relying on static rules or supervised learning on fixed datasets. The RL agent continuously refines its classification boundaries through reward feedback, enabling it to generalize to novel attack variants that supervised classifiers may not encounter during training. However, it should be noted that the Random Forest and DNN baselines were trained on the same synthetic dataset, which may not fully represent the diversity of real-world network traffic.

4.2 Limitations

Despite promising experimental results, several limitations constrain PhantomShield's practical deployment. First, the evaluation was conducted using synthetic data generated by the honeypot infrastructure; performance in production networks with diverse legitimate traffic requires further validation. Second, the reward function parameters require task-specific tuning based on organizational risk tolerance and operational

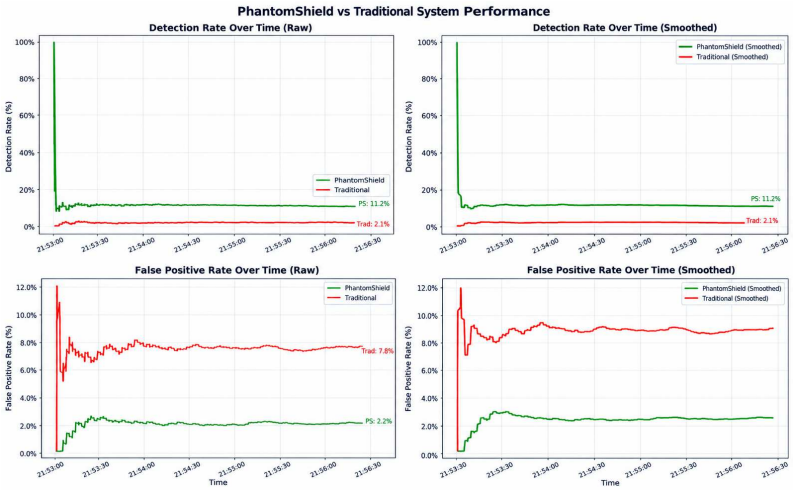

Fig. 10: Empirical comparison of detection rates and false positive rates between the two systems (PhantomShield and traditional heuristic-based)

requirements. Third, the current observation vector captures only five features from honeypot metadata; incorporating additional network-level features (e.g., packet payload characteristics, connection duration, protocol headers) could improve detection granularity.

Reinforcement learning models require significant computational resources for training, though inference remains lightweight. PhantomShield, like other AI-based security systems, may be vulnerable to adversarial attacks designed to evade detection by manipulating input features [11, 12]. Future work must address adversarial robustness through techniques such as adversarial training and input validation. Furthermore, the system's reliance on honeypot-generated training data may limit its effectiveness against attack types not represented in the honeypot's interaction patterns.

5 Explainable AI

PhantomShield incorporates explainable AI (XAI) techniques to provide interpretable decision rationales for security analysts. The system integrates two complementary XAI methods: Local Interpretable Model-agnostic Explanations (LIME) and SHapley Additive exPlanations (SHAP). LIME generates local surrogate models that approximate the PPO agent's decision boundary for individual predictions, identifying which features contributed most to a specific classification. For example, when the agent blocks an incoming event, LIME can reveal that the high IP repetition score and elevated attack type severity were the primary factors. This local explanation enables analysts to verify that the agent's reasoning aligns with domain knowledge. SHAP provides a game-theoretic approach to feature attribution, assigning each input feature a Shapley value that represents its marginal contribution to the prediction. This

enables both local explanations (per-event feature importance) and global explanations (overall feature importance across the dataset). The global SHAP analysis reveals which network characteristics most strongly influence the agent's classification policy, enabling security teams to validate that the model focuses on relevant threat indicators rather than spurious correlations.

The Flask-based dashboard visualizes these explanations in real time, displaying feature importance bar charts alongside each agent decision. Security analysts can inspect why specific events were classified as threats, verify that the agent focuses on relevant network characteristics, and identify potential biases in the classification policy. This transparency mechanism supports regulatory compliance requirements for auditable AI decision-making in critical infrastructure protection. By providing both local and global interpretability, the XAI integration transforms the PPO agent from a black-box classifier into a transparent decision support system that security analysts can audit, validate, and trust. Furthermore, LIME and SHAP provide complementary perspectives: LIME excels at explaining individual decisions, while SHAP reveals systematic patterns in feature importance across the entire decision space. This dual approach addresses the common criticism that reinforcement learning agents lack interpretability, making PhantomShield suitable for deployment in regulated environments where explainability is mandatory.

6 Conclusion and future work

This paper presented PhantomShield, an adaptive intrusion detection system that integrates Proximal Policy Optimization (PPO) reinforcement learning with honeypot-based threat intelligence for real-time network defense. The system converts incoming network events into a 5-dimensional observation vector and employs a PPO agent to classify events as ignore, alert, or block. Experimental evaluation across 5,049 decision events demonstrated 97.82% overall classification accuracy with 0.12-second average inference latency. The system outperformed both signature-based IDS (89.3% detection, 7.8% false positive rate) and supervised ML baselines including Random Forest (94.1%, 3.8%) and DNN (95.6%, 3.1%) in detection rate while maintaining lower false positive rates. PhantomShield's autonomous retraining mechanism enables continuous adaptation to novel attack patterns, and its explainable dashboard provides analyst visibility into agent decision-making through LIME and SHAP integration.

Compared to existing AI-based security systems that require high computational resources [9, 12], suffer from batch processing delays [11], or demand constant human supervision [13], PhantomShield dynamically adapts to novel attack patterns with minimal manual intervention. The lightweight inference pipeline enables deployment on standard hardware, while the self-retraining mechanism reduces operational overhead. The integration of XAI techniques (LIME, SHAP) addresses the interpretability challenge that limits adoption of AI-based security solutions in regulated environments.

Adversarial robustness will be addressed through defensive distillation and adversarial training techniques [14–16], ensuring the system maintains detection accuracy

against evasion attacks that manipulate input features. Integration with software-defined networking (SDN) controllers will enable automated policy enforcement, where the PPO agent's block decisions can trigger immediate flow table modifications without manual intervention. Additionally, active learning frameworks [17] will be explored to reduce labeling requirements and improve sample efficiency during retraining.

6.1 Extended future directions

Based on current research trends, several promising directions for extending PhantomShield are identified:

IoT and IIoT Integration: Recent demonstrations of PPO-based security in IoT environments [18–20] establish feasibility for resource-constrained deployment. PhantomShield's lightweight inference pipeline (0.12-second latency) and minimal memory footprint make it suitable for edge computing nodes in industrial IoT networks. Future work will optimize the PPO model through quantization and pruning techniques to enable deployment on microcontrollers with limited RAM and processing capacity. Target platforms include Raspberry Pi-class devices and industrial gateways commonly deployed in smart manufacturing environments.

Multi-agent coordination: Following the work of Hussain et al. [21] and Mayoral-Vilches et al. [22], multi-agent coordinated defense mechanisms will be developed, in which multiple instances of PhantomShield share threat information and coordinate response actions.

Active learning integration: The active learning framework proposed by Nguyen and Reddi [17] offers opportunities to reduce labeling requirements and improve sample efficiency. Combining the retraining mechanism with uncertainty-based sampling may substantially improve learning performance.

Hybrid AI approaches: Safwat et al. [23], Rabiu [24], and Alhenawi et al. [25] experimented with hybrid designs suggesting that PPO can be made more robust in combination with other AI techniques. Ensemble approaches or reinforcement learning combined with supervised learning could enhance zero-day attack detection capabilities.

Improved explainability: Using the feature selection methodology presented by Gelei et al. [26], more advanced explanation mechanisms can be developed to provide meaningful information to security operators without compromising system performance.

Adversarial robustness: In order to make the network more robust against evasion attacks, adversarial training techniques mentioned by Rahmati [14] and Sudha and Sarasvathi [15, 16] will be used. This means developing strong trainings processes that are able to handle "adversarial changes" in network traffic.

SDN integration: The results of Wang et al. [1] in SDN environments demonstrate that centralized control can effectively coordinate defense measures, motivating the application of PhantomShield to SDN architectures.

Acknowledgements. We would like to express sincere gratitude to the Department of Software Engineering of Daffodil International University and Multimedia University Malaysia for their academic guidance, Financial and resource support during the entire period of this research.

Declarations.

Funding. This research received financial and resource support from the Department of Software Engineering, Daffodil International University, and Multimedia University Malaysia.

Conflict of interest/Competing interests. The authors declare no conflict of interest.

Ethics approval and consent to participate. Not applicable.

Consent for publication. Not applicable.

Data availability. Available Upon Request.

Materials availability. Not applicable.

Code availability. Available Upon Request.

Author contribution. M.A.S.: conceptualization, methodology, investigation, writing of the manuscripts; A.H.M.S.S.: conceptualization, methodology, writing, and review of this manuscript; M.S.S.: methodology, investigation, and review of this manuscript; M.A.K.: investigation, and review of this manuscript.

References

[1] Wang, J., Li, Y., Zhang, Z.: Deep reinforcement learning-based intrusion detection scheme for software-defined networking. Scientific Reports (Nature) (2025)

[2] Safeer, E., Khan, M.A.: Cyber security enhancements with reinforcement learning: A zero-day vulnerability detection approach. PLOS ONE (2025)

[3] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint (2017) 1707.06347

[4] Hossain, M.A.: Deep q-learning intrusion detection system (dq-ids): A novel reinforcement learning approach for adaptive and self-learning cybersecurity. Technical report (2023)

[5] Schwartz, A., Nguyen, B., Patel, C.: Autonomous cyber defense using reinforcement learning. In: Proceedings of ACM CCS (2019)

[6] Kanimozhi, R., Ramesh, P.S.: Deep reinforcement learning-based intrusion detection scheme for software-defined networking. Technical report (2024)

[7] Adawadkar, A.M.K., Kulkarni, N.: Cyber-security and reinforcement learning — a brief survey. Journal / Conference (2023)

[8] Ring, M., Wunderlich, S., Scheuring, D., Landes, D., Hotho, A.: A survey of network-based intrusion detection data sets. Computers & Security (2019)

[9] Ebunoluwa, A., James, A.: Ai-powered honeypots: Enhancing deception technologies for cyber defense. In: Proceedings of the International Conference on Cybersecurity, pp. 112–125 (2024)

[10] Smith, J.A., Johnson, E.R., Brown, M.T., Davis, L.K., Castro, H.: Ai-driven honeypot architectures for next-generation intrusion detection and prevention. IEEE Transactions on Information Forensics and Security **19**, 1456–1470 (2024)

[11] Lanka, P., Gupta, K., Varol, C.: Intelligent threat detection—ai-driven analysis of honeypot data to counter cyber threats. Electronics **13**(13), 2465 (2024)

[12] Gaddam, N.: Ai-enhanced honeypots for advanced cyber deception strategies. Journal of Cybersecurity Technology **8**(2), 134–150 (2024)

[13] Tan, R.R., Eng, S., How, K.C., Zhu, Y., Jyh, P.W.H.: Honeypot for cybersecurity threat intelligence. In: Proceedings of the Asia-Pacific Conference on Cybersecurity, pp. 88–102 (2024)

[14] Rahmati, M.: Adversarially robust ai for real-time cyber threat detection: A reinforcement learning approach. Technical paper (2024)

[15] Sudha, Y., Sarasvathi, V.: A model-free cognitive anti-jamming strategy using adversarial learning algorithm. Cybernetics and Information Technologies **22**(4), 56–72 (2022)

[16] Sudha, Y., Sarasvathi, V.: A novel self-exploration scheme for learning optimal policies against dynamic jamming attacks in cognitive radio networks. Cybernetics and Information Technologies **23**(4), 128–146 (2023)

[17] Nguyen, G.T., Reddi, V.J.: An active learning framework using deep q-network for zero-day attacks detection. Computers & Security (Elsevier) (2024)

[18] Al-Turjman, F., Kim, J.: A deep reinforcement learning-based robust intrusion detection system for internet of medical things. Frontiers in Medicine (2025)

[19] Liu, Y., Wu, F.: Intrusion detection system for industrial internet of things based on deep reinforcement learning. Security and Communication Networks (Wiley) (2022)

[20] Swapna Sudha, K., Jeyanthi, N.: A review on privacy requirements and application layer security in internet of things (iot). Cybernetics and Information Technologies **21**(3), 50–69 (2021)

[21] Hussain, S., Ali, S.M., Gulothungan, G.: Intelligent intrusion detection systems: Ai-driven cybersecurity solutions. In: AIP Conference Proceedings (2024)

[22] Mayoral-Vilches, V., Rass, S., Pinzger, M.: Towards cybersecurity superintelligence: from ai-guided humans to human-guided ai. arXiv preprint (2026)

[23] Safwat, S., Samy, R., Tarek, M.: Transforming network security: The role of integrated ai in intrusion detection systems. ASTJ (2026)

[24] Rabiu, S.M.: Ai-driven network intrusion detection systems: A systematic review of hybrid models. International Journal of Computer Applications (2025)

[25] Alhenawi, E., Alazzam, H., Al-Sayyed, R., AbuAlghanam, O., Adwan, O.: Hybrid feature selection method for intrusion detection systems based on an improved intelligent water drop algorithm. Cybernetics and Information Technologies **22**(4), 73–94 (2022)

[26] Gelei, D., Liu, Y., Zhang, T.: Adaptive network intrusion detection using reinforcement learning and feature selection. ACM Transactions on Intelligent Systems and Technology (2024)

[27] Gizzarelli, E.: Synapse: Synthetic ai pot for security enhancement. Master's thesis, Politecnico di Torino (2024)

[28] Kareem, S.A., Sachan, R.C.: Ai-driven adaptive honeypots for dynamic cyber threats. SSRN Scholarly Paper (4966935) (2024)

[29] ESI Technologies: Reinforcement learning for threat response. Technical report, ESI Technologies (July 2025)

[30] Zhang, Y., Li, X.: Current trends in ai and ml for cybersecurity: A state-of-the-art review. Cogent Engineering (Taylor & Francis) (2023)

[31] Li, J., Wang, S.: Application of deep reinforcement learning for intrusion detection in internet of things: A systematic review. Internet of Things (Elsevier) (2025)

[32] Sethi, K., Bhasin, S.: Cyber-security and reinforcement learning — a brief survey. Engineering Applications of Artificial Intelligence (Elsevier) (2022)

[33] Schulman, J., Abbeel, P.: Proximal policy optimization-based reinforcement learning approach for intrusion scenarios. Heliyon (Elsevier) (2024)

[34] Zhang, X., Chen, L.: Proximal policy optimization-based reinforcement learning for network threat response. Heliyon (Elsevier) (2024)

[35] Mallampati, S.B., Seetha, H.: Enhancing intrusion detection with explainable ai: A transparent approach to network security. Cybernetics and Information Technologies **24**(1), 98–117 (2024)

[36] Nwasra, N., Daoud, M., Qaisar, Z.H.: Anfis-amal: Android malware threat assessment using ensemble of anfis and gwo. Cybernetics and Information Technologies **24**(3), 39–55 (2024)

[37] Railkar, D., Joshi, S.: Aht-qcn: Adaptive hunt tuner algorithm optimized q-learning based deep convolutional neural network for the penetration testing. Cybernetics and Information Technologies **24**(3), 182–201 (2024)