Pakistan Journal of Life and Social Sciences
www.pjlss.edu.pk

https://doi.org/10.57239/PJLSS-2026-24.1.0036

RESEARCH ARTICLE

Using Reinforcement Learning Algorithms for Dynamic Management of Security Policies

Andrey Aidinyan[1*], Petr Shevchuk[2]

[1,2] Don State Technical University, Rostov-on-Don, Russia

ARTICLE INFO

Received: APR 20, 2026

Accepted: MAY 18, 2026

Keywords

Reinforcement Learning
Dynamic Security Policy Management
Software-Defined Networking
Intrusion Response
Cyber Threat Mitigation

***Corresponding Author:**
andstyle@mail.ru

ABSTRACT

In the context of the rapid evolution of cyber threats, traditional static methods of managing security policies demonstrate limited adaptability and insufficient effectiveness against multi-vector attacks and zero-day threats. This paper proposes an approach to dynamic management of security policies based on Reinforcement Learning (RL). The management task is formalized as a Markov decision–making process (MDP), taking into account security constraints (CMDP) and the security-accessibility compromise. The architecture of the SDPM-RL agent system is presented, which is capable of modifying access rules and configurations of network security features in real time through the SDN execution loop. An experimental assessment in an emulated environment showed a 40% reduction in the average incident response time compared to static policies, while maintaining the level of false positives within acceptable limits and increasing the availability of services in the face of attacks. The results confirm the feasibility of integrating RL agents into the security management contours of modern information infrastructures.

INTRODUCTION

Modern information systems are characterized by high dynamism: topologies, node roles, traffic profiles, the composition of services and access policies are changing. At the same time, the threat model is becoming more complicated: attackers use multistep attack chains, low-intensity and "masking" techniques, and exploit the time window between threat detection and updating protection rules. Traditional security mechanisms (firewalls, ACLs, signature IDS/IPS) remain static in many scenarios: rules are set manually and updated discretely. Constant support by specialists is required for the correct configuration of such systems, which is problematic in large infrastructures.

The use of static policies leads to the fact that the policy is updated after damage is caused; blocking legitimate streams and reducing accessibility in order to reduce risk. Hence, there is a need to move to adaptive (self-adaptive) security, where policy changes are performed automatically based on the observed state of the system and threats. Reinforcement learning is a promising tool for this task, as it optimizes the sequence of decisions over time (response selection /policy adaptation) in conditions of uncertainty and incomplete observability, and not only classifies events.

The aim of the work is to develop an architecture and a method for dynamic management of security policies based on reinforcement learning and to evaluate the effectiveness of the approach based on emulated network attack scenarios.

LITERATURE REVIEW

The issue of automation of security policy management is traditionally considered through the prism of intrusion detection and prevention systems, as well as network filtering tools. It is emphasized in (Scarfone, Mell, 2007) that the effectiveness of IDS/IPS significantly depends on the

correct configuration, the quality of event sources and timely maintenance of rules, which in large-scale infrastructures leads to inevitable reaction delays and increased operating costs. At the same time, even with proper configuration, attackers actively bypass signature methods due to the variability of payloads and tactics, as well as the use of multistep attacks that are difficult to express with static rules. Additionally, it was shown in (Sommer, Paxson, 2010) that the transfer of ML methods to IDS scenarios is complicated by the drift of traffic distributions, the high cost of errors, and the possibility of purposefully bypassing the model, which does not achieve sustainable operational efficiency.

Against this background, considerable attention is being paid to the use of machine learning methods to detect anomalies and classify network events. In (Buczak, Guven, 2016), data mining and ML approaches for IDS tasks are systematized and it is demonstrated that models can increase sensitivity to unknown threats compared to signatures, however, in most cases they solve the problem of detection, rather than the problem of choosing the optimal management impact on the protection system. Similarly, in (Chandola et al., 2009), a general methodology for detecting anomalies is formed, but it does not consider decision-making on dynamic policy modification as a consistent optimization task. In (García-Teodoro et al., 2009), the key application difficulties of anomaly-based NIDS are described: resistance to changing load profiles, dependence on the feature description, as well as a compromise between sensitivity and the number of false alarms.

Another problem is that with a low proportion of real attacks in the total event stream, even a small number of false positives leads to the fact that most alarms turn out to be false, overloading operators and reducing the practical value of detection (Axelsson, 2000). The representativeness of the data remains an essential factor in the quality of ML solutions. In (Sharafaldin et al., 2018), the construction process and properties of the CICIDS2017 dataset are considered, which has become one of the most common benchmarks for evaluating IDS models, but by itself does not eliminate the problem of generalization to a "live" network with different policies and user behavior. In practical systems, lightweight online detection methods without full markup are also used, for example, Kitsune based on an ensemble of autoencoders, focused on processing streams at network gateways, which confirms the demand for adaptive methods in real time (Mirsky et al., 2018). Nevertheless, even with the high accuracy of the detector, the question remains which policy changes should be applied to minimize damage and at the same time not destroy the availability of services.

This formulation naturally leads to reinforcement learning as a paradigm for optimizing the sequence of actions in the environment. The fundamental results of Deep RL, including DQN, have shown that an agent can independently develop a strategy based on observations and scalar rewards, effectively working in complex state spaces (Mnih et al., 2015). Sustainability of learning is important for applied management tasks.; The PPO algorithm has become widespread due to stable optimization and good reproducibility in various environments, which makes it an attractive candidate for cybersecurity tasks where reliable policy convergence is required (Schulman et al., 2017). However, in cybersecurity, an agent should not perform actions that violate critical restrictions (for example, block management services or degrade accessibility to an unacceptable level). Therefore, the directions of Safe RL and optimization with constraints are significant. In (García, Fernández, 2015), approaches to safe training and control of agent actions are systematized, including penalty functions, action screening and optimization with risk constraints. In (Achiam et al., 2017), a Constrained Policy Optimization (CPO) was proposed, where the policy is optimized when specified constraints are met, which conceptually meets the requirements of a "safe" response in production infrastructures.

In relation to cybersecurity, reinforcement learning is considered as a mechanism for choosing countermeasures and dynamically managing defensive configurations. Reinforcement learning is especially promising where it is necessary to make decisions in conditions of uncertainty, incomplete observability and an adaptive opponent, that is, in tasks close to the actual operation of SOC (Nguyen, Reddi, 2023). Practice-oriented work on intrusion response demonstrates that model-free RL approaches can be trained to choose responses without an explicit attacker model, optimizing long-term consequences for the infrastructure (Hughes et al., 2022). In more recent studies, RL agents are trained to respond in realistic multistep scenarios. In (Reaney et al., 2024),

the authors showed the applicability of the DRL response in an IT-OT context (including taking into account the "destructiveness" of responses), and in (Iturbe et al., 2025), the authors investigated RL approaches for responding to advanced threats in realistic scenarios and compared the behavior of PPO and DQN.

A separate area of work is related to the protection of SDN networks. It is noted in (Kreutz et al., 2015) that the programmability of SDN provides a practical mechanism for implementing an adaptive security policy, but at the same time imposes requirements on the reliability of the controller and the correctness of the rules. The OpenFlow protocol described in (McKeown et al., 2008) provided a standard mechanism for setting flow processing rules through the controller, which makes SDN a natural "execution loop" for decisions made by an intelligent agent. It is the combination of "intelligent solution → fast programmable enforcement" that allows you to reduce response time and implement policy adaptation in real time.

At the same time, the introduction of ML/RL into security exacerbates the threats associated with the impact on models. Goodfellow et al. (2015) show the existence of adversarial examples capable of leading the model to erroneous decisions with small but specially selected input perturbations. Practical black-box attacks on models are demonstrated in (Papernot et al., 2017), which is especially important for scenarios where the attacker does not have access to the parameters, but is able to observe the system outputs. Gleave et al. (2020) show the possibility of "adversarial policies" attacks, when an adversary influences an agent through the dynamics of the environment, forcing him to make unfavorable decisions, and Wu et al. (2021) show methods of training attacking policies against RL systems (adversarial policy training), which makes the problem of the stability of RL protection to a strategic adversary especially relevant. Therefore, practical RL security systems should include mechanisms for limiting actions, monitoring and post-verification of decisions, as well as consider targeted circumvention scenarios.

Thus, an analysis of the literature shows a steady trend of transition from static policies and purely ML detection approaches to intelligent systems capable of making management decisions and adapting protection configurations in real time. At the same time, critical challenges remain: the correct formulation of the reward function, taking into account the availability and cost of changes, ensuring a safe RL/constraints, resistance to adversarial influences, and policy portability between different infrastructures. In the framework of this work, these aspects are taken into account through the formalization of the security policy management task as MDP/CMDP, the use of a stable PPO algorithm, the introduction of a composite reward function and the use of an SDN execution loop with action validation, which is aimed at practically reducing response time while maintaining acceptable service availability.

Problem Statement

For the application of reinforcement learning, the task of dynamic management of security policies is formalized as MDP (and expanded to CMDP in the presence of strict constraints), and is described by the set: $\langle S, A, P, R, \gamma \rangle$, where S is the state space of the system (the set of all possible network configurations); A is the agent's action space (the set of available control actions); P is a stochastic transition function unknown to the agent (depends on the attacker and legitimate users).

For the "security–accessibility–stability" balance, the composite form is used

$$P(s_{t+1} \mid s_t, a_t) = \Pr S_{t+1} = s_{t+1} \mid S_t = s_t, A_t = a_t .$$

R is a reward function that determines the immediate usefulness of an action in a given state: $S \times A \to R$, $R_t = R(s_t, a_t)$ — the discount coefficient determines the degree of influence of future rewards on the agent's current strategy.

The objective learning function is to maximize the expected discounted number of rewards: $G_t = R_t + \gamma \cdot R_{t+1} + \gamma^2 \cdot R_{t+2} + \cdots = \Sigma_{k=0}^{\infty} \gamma^k \cdot R_{t+k}$. The s_t state describes the observed configuration and the "risk level" in the network. Network status includes, for example, aggregated traffic statistics (intensity, port allocation, entropy of sources/destinations, SYN/ACK share, error

rate), node status indicators (availability, CPU/RAM load, service status), IDS signals (alert density by type/source per window), active indicators rules (hash or vector of features of a set of rules).

The actions A of the agent are interpreted as commands to change the policy, for example, blocking the IP/subnet for a specified time (TTL), rate-limit for the stream/port, redirecting part of the traffic to the honeypot or mirror, strengthening logging/inspection for the selected service, segment isolation (micro-segmentation). A composite form is used to ensure the security–accessibility–stability balance. The reward function in step t is calculated using the formula

$$R_t = w_{sec} \cdot r_{sec}(s_t, a_t) + w_{avail} \cdot r_{avail}(s_t, a_t) + w_{stab} \cdot r_{stab}(s_t, a_t),$$

where w_{sec}, w_{avail}, w_{stab} — weighting factors of component importance; r_{sec} is a component reflecting the security level.; r_{avail} is a component that reflects the availability of services.; r_{stab} is a component that reflects the stability of the system. The goal is to find the $\pi(a \mid s)$ policy that maximizes the expected discounted number of rewards:

$$\max_{\pi}\left(\sum_{t=0}^{\infty}\gamma^t R_t\right).$$

Proposed Architecture of the Sdpm-Rl System

The architecture of the SDPM-RL (Security Dynamic Policy Management with Reinforcement Learning) dynamic security Policy Management system is built as a closed loop "observation → decision-making → execution → result control", focused on automating the response to cyber incidents while maintaining operational manageability. The architecture is based on the idea of separating functional roles: collecting and preparing telemetry, choosing a countermeasure based on a learnable policy, and deterministically applying solutions through a programmable execution loop. This decomposition allows you to simultaneously ensure adaptability through reinforcement learning and reduce the risk of disruptive actions due to built-in constraints and validation.

At the entrance of the system there is a monitoring module, which receives monitoring data from the protected infrastructure: aggregated network traffic statistics (for example, flow parameters, port distribution, entropy of sources/destinations, SYN/ACK ratio, error rates), events and metrics of hosts (service availability, CPU/RAM utilization, status network interfaces), as well as IDS/IPS alerts and security logs. Since the use of "raw" packets makes the task excessively high-dimensional and worsens the output delay, the telemetry is converted into a compact feature vector. At this stage, normalization, smoothing, and aggregation are performed, as well as the formation of historical features (for example, moving averages or differences), which allows you to take into account the dynamics of threats and reduces sensitivity to short-term spikes in legitimate load. The generated state s_t is passed to the RL agent, which implements the policy $\pi(a \mid s)$and selects the action a_t from the specified space of countermeasures.

The basic algorithm uses Proximal Policy Optimization (PPO), because it provides sustained policy optimization and performs well in stochastic environments typical of network security. The agent includes two components: an actor who forms a probability distribution over actions (and, if necessary, over the parameters of actions), and a critic who evaluates the value of the state to reduce the variance of the gradient and increase the stability of learning. It is fundamentally important that the agent optimizes not the local detection metric, but the long-term reward set by a composite reward function that balances security, service availability, and configuration stability. This allows you to avoid the "aggressive" strategy of total blocking, keeping services operational in the face of attacks. Agent training is organized according to a hybrid scheme that minimizes risks to productive infrastructure, consisting of two stages: Pre-training and Fine-tuning. At the first stage, preliminary training is performed in an isolated simulated environment (cyberpolygon/emulation), where typical attack scenarios and historical traffic profiles are reproduced. At the second stage, additional training is performed in an environment close to the real one, with a limited degree of exploration and using safety mechanisms that prevent critical actions during the adaptation stage. This approach allows an agent to acquire basic skills without the risk of malfunction, and then carefully adjust to the specifics of a particular infrastructure and its traffic.

The agent's solution passes through the secure execution layer before application, which performs the "action shielding" function and checks the action for compliance with the basic constraints. Validation includes prohibitions on blocking critical addresses and management services, control of rule conflicts, limiting the frequency of policy updates, as well as checking acceptable parameter ranges (for example, minimum rate-limit values, maximum TTL locks). This layer plays the role of a fuse: even if an agent in a rare case suggests an undesirable action, the system will not allow its use and will preserve the manageability of the infrastructure. After passing the validation, the action is translated into configuration commands for network and security components.

SDPM-RL uses the SDN approach as the execution loop: the agent generates requests through the controller's northern API, and the controller converts them into flow processing rules (for example, flow table entries for OpenFlow switches) or into microsegmentation policies. The centralized nature of SDN management ensures fast and consistent enforcement over the network, which is critical for reducing incident response time. In parallel, the system can integrate with traditional tools (IDS/IPS, FW, WAF) through their control interfaces, however, the SDN component allows you to implement dynamic network policy at the flow level. The closed loop is ensured by the fact that after applying the action, the system continues to monitor the state of the infrastructure and assesses the effect of the applied measures through changes in features and operational metrics. This data is used to calculate the reward and, if necessary, to further train the model. Thus, SDPM-RL implements adaptive management of security policies in real time, combining intelligent RL-based decision-making with a programmable and controlled execution mechanism, which allows for accelerated response while maintaining acceptable availability and manageability.

Figure 1 shows an enlarged closed-loop diagram of the dynamic management of SDPM-RL security policies. The "Monitoring" block is shown on the left, which aggregates telemetry from monitoring sources (network traffic analysis, IDS alerts, system logs) and generates the s_t status vector. The "Agent RL" block implements the policy $\pi(a \mid s)$ with the help of an actor and the assessment of the value of $V(s)$ with the help of a critic; based on the condition, the action a_t is calculated, and the reward function $R(\cdot)$ is used, reflecting the compromise between security, availability and stability of changes. The "Execution" block is located on the right. The execution module translates the agent's decisions into configuration commands: the RL agent sends actions to the SDN controller via the Northbound API, and the controller converts the decision into rules for switches or microsegmentation policies, and thereby controls the equipment via the Southbound API (for example, OpenFlow). The execution module includes a mechanism for validating actions, after which the control action is translated into policy updates at the security level (firewalls, switches/SDN components, IDS). The action validation mechanism prohibits blocking of "critical" addresses/ports (management plane, DNS, monitoring), limits the frequency of changes and the lifetime of locks, and checks for rule conflicts (policy sanity checks).

The lower part of the diagram reflects the learning contour: pre-training in a cyberpolygon and subsequent fine-tuning during deployment, which reduces risks in a productive environment. The closing feedback arrow indicates that the results of policy application are returned to the telemetry and used for subsequent agent decisions.

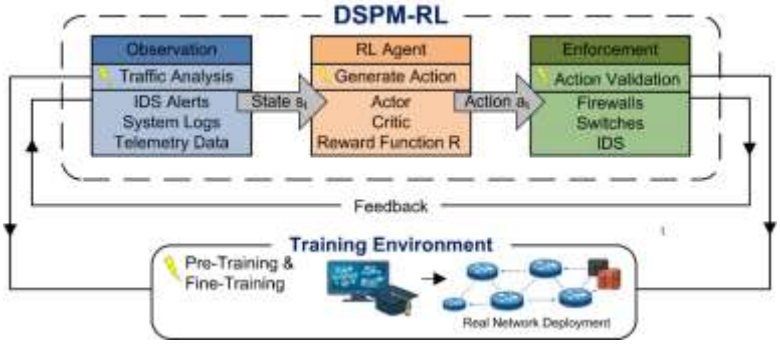

Figure 1: Architecture of the SDPM-RL system

Description of the Experiment

The SDPM-RL performance test bench was deployed in an emulated network environment that reproduces a typical corporate segment with several classes of nodes and services. Mininet was used as the basis, which allows modeling the network topology, nodes and communication channels with controlled parameters. Mininet is a network emulator that allows you to quickly deploy virtual topologies of hosts, switches, and communication channels on a single host using virtualization mechanisms at the OS level (processes, network namespaces, virtual Ethernet), while running the "real" Linux network stack and application code without modification, which makes the environment convenient for reproducible SDNs-experiments and testing of security policies (Lantz et al., 2010).

Within the topology, server components (web server and database server), user workstations, and infrastructure elements forming a characteristic client–server traffic profile were identified. Network policies were managed and applied operationally through an SDN loop: the controller was implemented on the basis of Ryu and was used to set flow processing rules on switches (in the OpenFlow/flow table paradigm), which provided rapid application of countermeasures and the possibility of software modification of routing and filtering. The Snort intrusion detection system in IDS mode was used to generate threat signals and receive security events; its alerts were used as one of the sources of status signs, together with traffic statistics and system logs. The RL agent is implemented in Python using the Stable-Baselines3 library, selected for a reproducible implementation of PPO and convenient integration with the user interaction environment. Training and experiments were performed on a dedicated computing server with an NVIDIA Tesla V100 GPU, which made it possible to accelerate the learning stages of neural network policy and conduct a series of runs with repeatable parameters. To reproduce the attacking effects, security testing tools (including load generation tools and typical operating frameworks) were used, which provided DDoS load modeling, port scanning, and credential enumeration in controlled emulation.

Figure 2 shows the organization of an experimental stand used to simulate attacks and test the effectiveness of dynamic security policy management. The upper-left part presents external threats – a toolkit-based attack traffic generator (Metasploit/LOIC) that generates the load and attack scenarios. In the upper right part, a control plane is highlighted, including the Ryu SDN controller, which uses the OpenFlow interface to set the rules for processing flows on the switch in an emulated network. The central area reflects the Mininet emulation environment, in which the main components of the enterprise segment are deployed: a web server, workstations, and an OpenFlow switch through which client-server and attacking traffic passes. The lower part of the circuit corresponds to an intelligent module: an RL agent (PPO) implemented in Python/Stable-Baselines3 and performing calculations on the GPU receives telemetry (network statistics and/or alerts) from the emulation environment and uses it to select control actions. The selected solutions are translated into network policy changes via the SDN loop (controller → switch), closing the "observation – solution – application" cycle, which was used to evaluate metrics (reaction time,). false locks, service availability

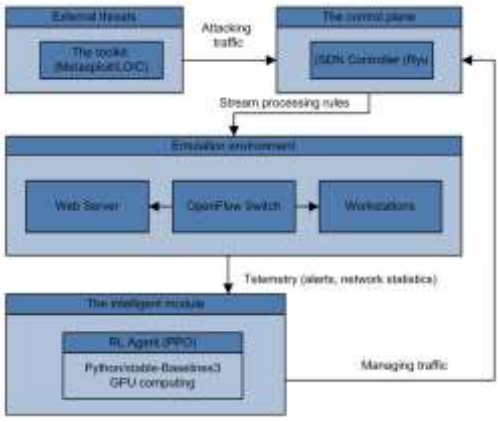

Figure 2: Block diagram of the experimental test bench for SDPM-RL evaluation

RESULTS AND DISCUSSION

Experimental tests have shown that the proposed SDPM-RL system forms a stable response strategy and surpasses static rules in terms of the speed of taking and executing countermeasures. During the pre-training in a simulated environment, there was an increase in average remuneration and subsequent stabilization, indicating the formation of a coordinated policy. At the same time, the tasks of cybersecurity are characterized by a stochastic and unstable environment, therefore, an important indicator of quality is not a "perfectly smooth" reward curve, but the absence of degradation of operational metrics (availability, frequency of rule changes, the proportion of false locks) after reaching a plateau. The key factor in the stability of behavior was the composite reward function and, in particular, the penalty for frequent configuration changes C^{change}. Its inclusion reduced excessive policy switching in response to minor traffic fluctuations — and made the policy more inertial. That is, the changes were applied mainly when there were persistent signs of an attack. Additionally, stability was provided by a mechanism for validating actions in the execution module, which prevented potentially dangerous operations (for example, blocking critical management services) and thereby reduced the risk of destructive agent errors.

During testing under conditions of simulation of intense attacks, the basic configuration ensured the availability of services at the level of 85.4%. The implementation of the adaptive agent has increased this indicator to 94.1%, which corresponds to a reduction in downtime by more than half. In parallel, there was an improvement in response time from 120.5 to 72.3 seconds (–40%) and an increase in the success rate of DDoS attacks from 60% to 92%. At the same time, the increase in the proportion of false positive positives (FPR) remained within the acceptable range (from 1.2% to 1.5%). Table 1 shows a comparison of the effectiveness of SDPM-RL and static rules.

Table 1: Comparison of the effectiveness of SDPM-RL and static rules

Metric	Static Rules (Baseline)	SDPM-RL	Absolute Δ	Relative Δ (%)
Mean Time to Mitigation (s)	120.5	72.3	–48,2 s	–40.0%
FPR (%)	1.2	1.5	+0.3 pp	+25,0%
Service Availability (%)	85.4	94.1	+8.7 pp	+10,2%
DDoS Mitigation Success Rate (%)	60.0	92.0	+32.0 pp	+53,3%

It can be seen that SDPM-RL provides reduced incident response time (TTM) and increased service availability in the face of attacks, while achieving a significant increase in the success rate of DDoS reflection. A slight increase in FPR is considered as an acceptable compromise, compensated by an increase in the overall stability of services and the availability of secure execution mechanisms (validation of actions, limitation of the frequency of changes and temporary locks). From the results obtained, it can be concluded that:

- The RL approach significantly reduces reaction time, as decisions are made automatically based on telemetry, without delays in manual analysis.

- The greatest gain is observed in DDoS scenarios, where the agent learns to quickly limit sources and redistribute flows, minimizing service degradation.

- A small increase in FPR is an expected compromise under attack conditions and is offset by increased availability of critical services.

The results confirm that reinforcement learning can be a practical mechanism for moving from static security policies to adaptive management, especially in environments with high traffic dynamics and frequent threat changes. The key advantage of SDPM-RL is that the agent does not solve the task of "detecting an attack" as such, but rather the task of "choosing an action" that minimizes long-term damage to the infrastructure, taking into account operational constraints. This fundamentally distinguishes the approach from most ML-IDS solutions, which are limited to event classification and further require manual decision-making by the operator, which increases delays and creates response variability. SDN allows you to centrally and quickly change the rules

for processing streams, which reduces the time gap between the identification of threat indicators and the actual use of countermeasures. In this sense, the RL agent becomes an intelligent policy module, and the SDN becomes a deterministic execution tool, which together forms a closed control loop.

In practical implementation, it is necessary to take into account the threats to the RL system itself. Modern research shows that ML models can be subject to adversarial influences, and RL agents can be vulnerable to strategic opponents influencing the dynamics of the environment. Therefore, the effectiveness assessment should include not only the "usual" scenarios of attacks on the infrastructure, but also scenarios where the attacker is trying to manipulate surveillance signals and indirectly influence the agent's choice of actions. At the design level, this means the need to combine RL with robust/safe methods, monitoring of feature distributions (drift detection), periodic retraining, as well as using a "digital twin" or an isolated environment to test policy updates before applying them in a productive loop. The experimental results were obtained in an emulated environment, which imposes limitations on the validity of the conclusions. Despite the use of typical attacks and realistic components (SDN controller, IDS), emulation does not fully reproduce the complexity of real networks, user behavior, quality of telemetry and a variety of attacking techniques. To confirm the portability of the results, additional tests in the cyberpolygon and pilot deployment on a limited segment of the real infrastructure are needed, with a gradual expansion of the scope and constant monitoring of operational risks.

CONCLUSION

The paper considers the application of reinforcement learning algorithms for the dynamic management of security policies. The proposed SDPM-RL architecture combines an RL agent (PPO), a composite reward function, and an SDN execution loop with action validation. Experiments in emulation have confirmed that the approach allows you to speed up the response to incidents and increase the stability of services in the face of attacks compared to static rules, while maintaining an acceptable level of false positive locks.

REFERENCES

Achiam J, Held D, Tamar A, Abbeel P. Constrained policy optimization. In: Proceedings of the 34th International Conference on Machine Learning (ICML). Proceedings of Machine Learning Research; 2017. p. 22-31; also available as arXiv:1705.10528.

Axelsson S. The base-rate fallacy and the difficulty of intrusion detection. ACM Trans Inf Syst Secur 2000,3:186-205. https://doi.org/10.1145/357830.357849

Buczak AL, Guven E. A survey of data mining and machine learning methods for cyber security Intrusion detection. IEEE Commun Surv Tutor 2016,18:1153-76. https://doi.org/10.1109/COMST.2015.2494502

Chandola V, Banerjee A, Kumar V. Anomaly detection: A survey. ACM Comput Surv 2009,41:15. https://doi.org/10.1145/1541880.1541882

García J, Fernández F. A comprehensive survey on safe reinforcement learning. J Mach Learn Res 2015,16:1437-80.

García-Teodoro P, Díaz-Verdejo J, Maciá-Fernández G, Vázquez E. Anomaly-based network intrusion detection: Techniques, systems and challenges. Comput Secur 2009,28:18-28. https://doi.org/10.1016/j.cose.2008.08.003

Gleave A, Dennis M, Wild C, Kant N, Levine S, Russell S. Adversarial policies: Attacking deep reinforcement learning. In: Proceedings of the 8th International Conference on Learning Representations (ICLR 2020). Red Hook, NY: Curran Associates, Inc.; 2020; also available as arXiv:1905.1061.

Goodfellow IJ, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. In: Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015). Red Hook, NY: Curran Associates, Inc.; 2015; also available as arXiv:1412.6572.

Hughes K, McLaughlin K, Sezer S. A model-free approach to intrusion response systems. J Inf Secur Appl 2022,66:103150. https://doi.org/10.1016/j.jisa.2022.103150

Iturbe E, Rego A, Llorente-Vazquez O, Rios E, Dalamagkas C, Merkouris D, Toledo N. Reinforcement learning in action: Powering intelligent intrusion responses to advanced cyber threats in realistic scenarios. Expert Syst Appl 2025,296:129168. https://doi.org/10.1016/j.eswa.2025.129168

Kreutz D, Ramos FMV, Verissimo P, Rothenberg CE, Azodolmolky S, Uhlig S. Software-defined networking: A comprehensive survey. Proc IEEE 2015,103:14-76. https://doi.org/10.1109/JPROC.2014.2371999

Lantz B, Heller B, McKeown N. A network in a laptop: Rapid prototyping for software-defined networks. In: HotNets-IX: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, Monterey, CA, USA, October 20-21, 2010. New York, NY: Association for Computing Machinery; 2010. art. 19. https://doi.org/10.1145/1868447.1868466

McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Shenker S, Turner J. Openflow: Enabling innovation in campus networks. ACM SIGCOMM Comput Commun Rev 2008,38:69-74. https://doi.org/10.1145/1355734.1355746

Mirsky Y, Doitshman T, Elovici Y, Shabtai A. Kitsune: An ensemble of autoencoders for online network intrusion detection. In: Proceedings of the Network and Distributed System Security Symposium (NDSS). The Internet Society; 2018. https://doi.org/10.14722/ndss.2018.23211

Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D. Human-level control through deep reinforcement learning. Nature 2015,518:529-33. https://doi.org/10.1038/nature14236

Nguyen TT, Reddi VJ. Deep reinforcement learning for cyber security. IEEE Trans Neural Netw Learn Syst 2023,34:3779-95. https://doi.org/10.1109/TNNLS.2021.3121870

Papernot N, McDaniel P, Goodfellow I, Jha S, Celik ZB, Swami A. Practical black-box attacks against machine learning. In: ASIA CCS '17: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security. New York, NY: Association for Computing Machinery; 2017. p. 506-19. https://doi.org/10.1145/3052973.3053009

Reaney M, McLaughlin K, Grant J. Network intrusion response using deep reinforcement learning in an aircraft IT-OT scenario. In: ARES 2024: Proceedings of the 19th International Conference on Availability, Reliability and Security. New York, NY: Association for Computing Machinery; 2024. art. 51. https://doi.org/10.1145/3664476.3670917

Scarfone K, Mell P. Guide to intrusion detection and prevention systems (IDPS). NIST Special Publication 800-94. Gaithersburg, MD: National Institute of Standards and Technology; 2007. https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=50951

Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. arXiv:1707.06347 [cs.LG]; 2017.

Sharafaldin I, Habibi Lashkari A, Ghorbani AA. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP), Funchal, Madeira, Portugal, 22-24 January 2018. Setúbal: SciTePress; 2018. p. 108-18. https://doi.org/10.5220/0006639801080116

Sommer R, Paxson V. Outside the closed world: On using machine learning for network intrusion detection. In: IEEE Symposium on Security and Privacy (IEEE, 2010). New York, NY: IEEE; 2010. p. 305-16. https://doi.org/10.1109/SP.2010.25

Wu X, Guo W, Wei H, Xing X. Adversarial policy training against deep reinforcement learning. In: Proceedings of the 30th USENIX Security Symposium, August 11–13, 2021. Berkeley, CA: USENIX Association; 2021. p. 1883-900.