Adaptive Firewall Strategy Generation and Optimization Based on Reinforcement Learning

Minghong Yang[1], Zhenhua Yang[1,*], Qiwen Yang[2]
[1]School of Economics and Management, Hunan Applied Technology University, Changde, Hunan,415100, China
[2]School of Computer Science, Beijing University of Aeronautics and Astronautics, Beijing ,100000, China
E-mail: yangzhenhua1988@hotmail.com
[*]Corresponding author

Keywords: adaptive firewall, reinforcement learning, deep Q-Learning, network security, policy optimization, markov decision process (MDP)

Received: May 23, 2025

Traditional firewall systems use static rule sets, making them unsuitable for growing cyber threats and network circumstances. In this research, we automate firewall rule development and optimization using reinforcement learning (RL) to increase network security and reduce human setup. This paper introduces FireRL, a system that uses reinforcement learning to help make smart decisions about firewall rules by treating it like a Markov Decision Process, enabling an RL agent to learn good firewall rules from simulated network traffic. This proposed method utilizes the Deep Q-learning algorithm to balance throughput, latency, and threat mitigation via repeated improvement. Experiments are performed in benign and dangerous traffic situations. Firewalls outperform static firewalls because they quickly react to new threats and reduce false positives. Resistance to new attack vectors demonstrates the system's flexibility and resilience. This research concludes with a self-optimizing firewall approach that greatly lowers expert-led settings. FireRL's proactive and scalable RL-based defense is ideal for current cybersecurity.

Povzetek: Članek združuje digitalne dvojčke, federativno učenje in blockchain v sistem za zaznavanje IoT groženj, ki izboljša varnost, zasebnost in učinkovitost.

1 Introduction

Cyberattacks are more numerous, sophisticated, and devastating than ever. As mission-critical systems like corporate networks, cloud platforms, and IoT ecosystems grow in size and complexity, digital infrastructure security becomes increasingly important. Network security depends on firewalls, which restrict incoming and outgoing data traffic based on rules. Modern attacks are nimble, sneaky, and unexpected, making conventional firewalls worthless. Such firewalls use static rule sets manually created by security specialists. Static rule-based systems are slow to adapt to new attack vectors, readily misconfigured, and lack contextual awareness for real-time behavior monitoring. [1] say Next Generation Firewalls (NGFWs) depend too much on static rules and human involvement to tackle dynamic cyber threats. This applies even with application-layer filtering and deep packet inspection in NGFWs. Since rules are harder to monitor and update in quickly changing settings, misconfiguration is more probable, resulting in security problems or network performance issues.

Many individuals have presented solutions for classic firewall difficulties [2] demonstrated unexpected traffic conditions by modeling packet-filtering firewalls using neutrosophic Petri nets. This method is ordered and logical. Despite their theoretical validity, these models cannot learn and adapt. [3] suggested integrating firewalls with IDS to improve detection. This hybrid method detects risks better, but overly strict policy definitions remain. Security systems are changing rapidly owing to machine learning (ML) and artificial intelligence (AI). ML has performed well in Web Application Firewalls (WAFs). [4] detected online threats better than standard WAFs using feature engineering and supervised learning. [5] improved complicated web-based threat detection using deep learning. These models are passive detectors that do not create or improve firewall rules.

The author investigated WAF anomaly detection using deep learning. Their technology can detect strange traffic patterns but can't adjust firewalls to attacks [6]. [7] also suggest automating virtualized firewall settings. Administrators can predefine templates, but their system can't automatically adjust to traffic circumstances. In IoT, adaptive, clever, and fast-reacting security solutions are in demand. Without a flexible firewall, devices in these networks are easy prey for attackers due to their low processing capability. [8]

highlighted the significance of segmentation and effective firewalling in safeguarding Internet of Things ecosystems. However, their solution is vulnerable to new threats since it is based on rule sets that human developed. This inquiry was also motivated by the increasing popularity of predictive security systems, which served as a source of inspiration. [9] provided a real-time anomaly detection technique for smart systems based on deep belief networks. This was done in acknowledgment that it is necessary to recognize outliers and potential threats because of the nature of the situation. The findings of their study indicate that learning algorithms, as opposed to static anomaly levels or preset signatures, should be used by firewalls to make prediction judgments.

To address these issues, this paper proposes FireRL, a real-time firewall architecture that learns and decides. Key points of this work:

- Design the proposed FireRL architecture to formalize firewall rule optimization in a reinforcement learning framework, enabling the agent to learn optimal tactics from its environment.
- FireRL introduces intelligent, autonomous, and flexible policy generation, shifting firewall design away from static, expert-driven rule management. These advances enable next-generation cybersecurity systems that foresee, prevent, and react to threats.
- Reward systems let the RL agent balance security enforcement (stopping malicious traffic) with performance measurements (latency, throughput).
- The recommended architecture is more flexible, generates fewer false positives, and has greater generalization on network traffic like zero-day assaults and adversarial behaviors.

2 Literature method

Current cybersecurity frameworks indicate that automation, deep learning, and artificial intelligence improve firewall performance in many computer systems. Although their slow response time and restrictive architecture prevent them from keeping pace with evolving threats, traditional firewalls remain essential for network security. The literature reflects a rising trend toward smart hybrid solutions that are also policy-aware in response to these limitations. Table 1 shows the comparative analysis of existing methods.

Table 1: Comparative analysis of existing methods

Ref. No.	Reference	Approach	Strengths	Limitations
[10]	Ekhlas Kadhim Hamza et al. (2024)	Used PPO, UCB, and Epsilon-Greedy RL algorithms for adaptive load balancing in publish/subscribe systems.	Improved message delivery, reduced latency, and adaptive decision-making.	PPO is resource-intensive; UCB and Epsilon-Greedy may underperform in complex environments.
[11]	Sepczuk, M. (2023)	Cyber Mimic Defense-Based WAF (CMD-BWAF)	Dynamic deception and mimicry to mislead attackers; real-time adaptability	High complexity; increased resource consumption; sensitive to tuning
[12]	Rajasoundaran, S. et al. (2024)	Energy-Proficient Firewall Policies for Wireless Networks	Energy-saving techniques for resource-constrained devices	Trade-off between energy efficiency and security strength; limited real-time learning
[13]	Lee, J. K. et al. (2024)	AI-Based Firewall Rule Refinement in HPC Networks	Scalable AI-based rule optimization reduces conflicts and redundancy	No adaptive decision-making; reactive rather than proactive optimization
[14]	Leka, E. et al. (2024)	ML and Blockchain-Based WAF (ML-BC-BWAF) for DDoS Mitigation	Tamper-resistant logs and enhanced anomaly detection	High system overhead; latency due to blockchain validation

[15]	Fadlil, A. et al. (2024)	SQL Injection Mitigation using Open Web Application Security Project (OWASP) Framework	Effective application-layer protection against SQLi	Not adaptable to new types of attacks; lacks a network-level defense
[16]	Kouassi, T. et al. (2024)	Security Policy Model in Hybrid TOGAF-Zachman Framework	Structured and formal policy management in enterprise cloud setups	The theoretical model lacks practical implementation and real-time enforcement capabilities

3 Proposed methodology

This paper presents FireRL, which uses reinforcement learning to provide adaptive firewall regulation methods. Its major aim is to adapt to changing network threats. Since static rules are obsolete, traditional firewalls can't respond to attacks in real time or account for new vectors. FireRL simulates an MDP to overcome these limits and choose the appropriate action given the network state. This technology allows the system to learn from its surroundings and alter its protective rules in real time to decrease dangers, delays, policy disobedience, and false positives. The agent employs a Deep Q-Network (DQN) to learn in high-dimensional state spaces to find the optimum Q-values for each state-action combination. The recommended method prioritizes experience replay to ensure convergence by emphasizing learning from significant or unexpected events. For FireRL's learning policy to operate in real time, its multi-objective reward function must balance threat detection and network efficiency. This can be game-changing. Using policy vector encoding, the system might grow while conserving resources. This allows several rules to be activated even with minimal computational resources. FireRL is a self-optimizing firewall that evolves its rule set. This secures complex networks and improves operations.

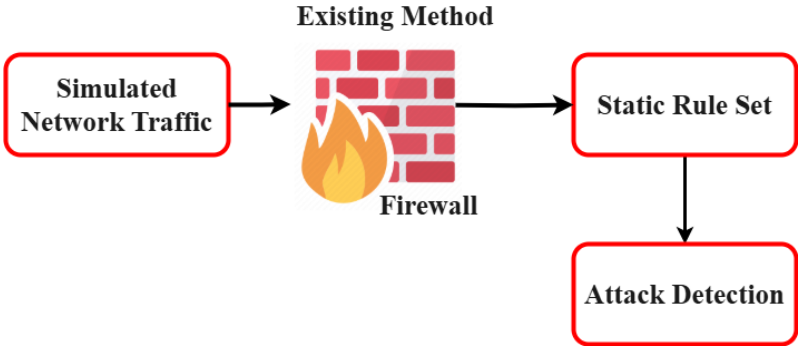

Figure 1: Existing methods-based firewall strategy

Figure 1 above depicts the "Existing Method," the standard intrusion detection method that uses firewalls. The system is subjected to simulated network traffic, beginning with benign and malicious data. Before selecting whether or not to allow incoming data to get through, firewalls evaluate it based on a set of predetermined policies and standards for security and filtering. Static rule sets, unlike dynamic rule sets, do not include any established criteria in their decision-making process on whether to accept or deny traffic. Immediately after verifying the data to ensure that it is in accordance with these standards, the system moves on to the attack detection phase, where it compares the data to the criteria to identify any potential threats. Because the technique is based on preset criteria, which cannot adapt to changing cyber threats or traffic patterns, it may have difficulty identifying new attacks or responding to unexpected surroundings. Conventional firewalls are characterized by their rule-based architecture, linear operation, and reliance on established criteria for threat detection, as shown by the image.

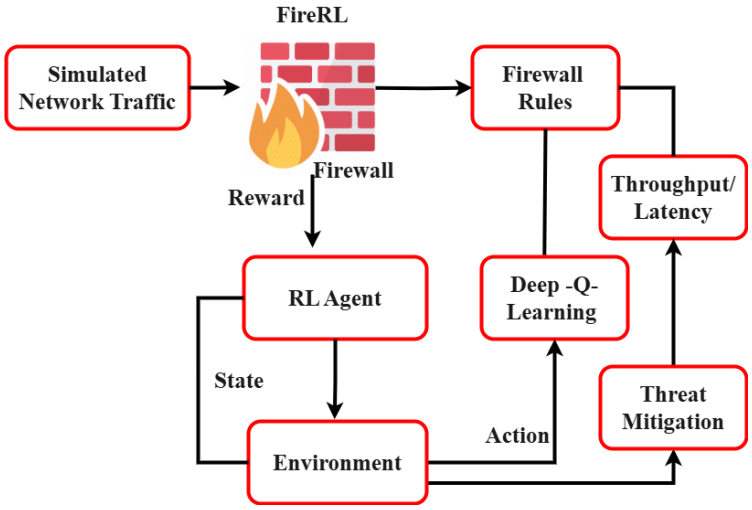

Figure 2: Proposed diagram

The FireRL framework is an intelligent firewall system that uses reinforcement learning to overcome network security restrictions dynamically. Figure 2 depicts a schematic of the FireRL framework's architectural approach. When regulating and filtering incoming data, the first step is to evaluate and implement firewall rules using the FireRL firewall. Simulating real network activity is the method by which it accomplishes this goal. An RL agent can increase its performance by generating a reward signal and integrating the firewall's findings with the threats it has discovered. An RL agent receives this signal as it is being sent. Real-life agents (RL) are responsible for monitoring their state and responding correctly based on their observations to interact with the network, regardless of whether it is virtual or actual. Using a Deep Q-Learning technique, the agent acquires knowledge about the most effective firewall rule regulations to simplify these processes. Latency and throughput on the network are two metrics used to assess the degree to which these activities successfully reduce risks.

As more time passes, the FireRL system improves its ability to make decisions by continually modifying the firewall rules to establish a balance between the two competing priorities of security and performance. The architecture ensures the firewall can constantly adapt to varied network circumstances and new threats, providing proactive and intelligent network security. In a word, the design ensures that the firewall is always effective.

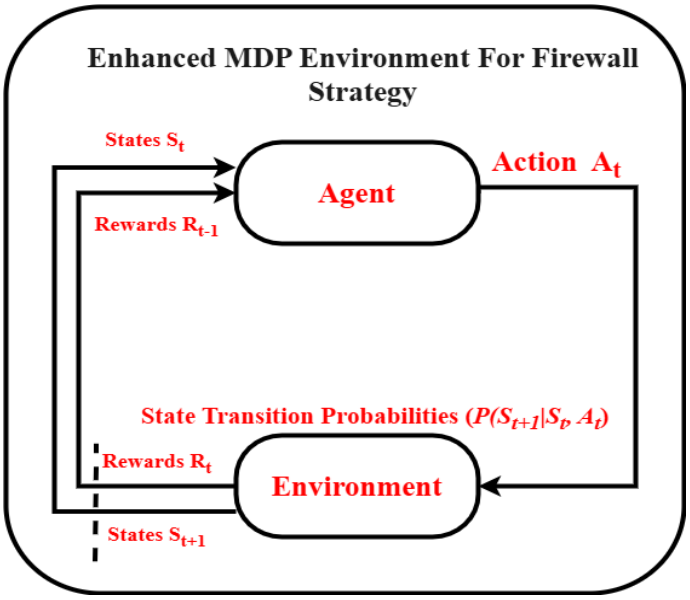

Figure 3: Enhanced MDP environment for firewall strategy

Figure 3 shows that within the context of improving dynamic firewall management, the graphic named "Enhanced MDP Environment for Firewall Strategy" illustrates a reinforcement learning strategy that uses Markov Decision Process (MDP). Within the framework of this architecture, the agent has the responsibility of implementing learnt rules (π) in response to an environment, to identify the most suitable firewall techniques. At each time step t, the agent decides on what actions to take, such as adding, altering, or deleting firewall rules, depending on the current state of the network environment $\boldsymbol{s_t}$. As a consequence of this action, a new state, denoted by s_{t+1}, is introduced into the environment. Additionally, a reward, denoted by $\boldsymbol{r_t}$, is allocated to the environment based on the probability $\boldsymbol{P(s'|s,a)}$, which shows how the action successfully maintained the network's security and operational efficiency.

Because the agent's policy may be modified over time in response to changes in the surrounding environment, the reward function $R(s,a)$ evaluates the effect of the action. Using this feedback loop, the agent can continually change its behaviors to increase its accumulated rewards. As a result of a discount factor γ, which may have values ranging from 0 to 1, the importance of profits in the near term is greater than the significance of gains in the long run. Not only does the system improve real-time danger awareness, but it can also respond to environmental changes by sending out alerts or warnings. The system can react to potential dangers as a result of this. This agent provides an alternative to static rule-based techniques since it can dynamically adapt firewall settings in response to changing network conditions and attack patterns.

$$M = (S = \{s_1, s_2, \dots, s_m\}, A = \{a_1, a_2, \dots, a_m\}. P(s'|s,a) = Pr[s_{t+1} = s'|s_t = s, a_t = s], R: S \times A \to R, \gamma \in [0,1] \quad (1)$$

As equation (1) was calculated, the enhanced MDRP environment definition was deliberated. A mathematical framework, the Markov Decision Process (MDP), depicts the FireRL system. This strategy makes it possible to make decisions sequentially when confronted with uncertainty. Given the present value of S, there is an infinite number of possible states that might be achieved by the network. Some examples of states that might be present include the use of resources, the configuration of firewalls, the attempts to break into the network, and the traffic patterns that are currently in place. The letter a represents a variety of possible RL agent operations that may be put into effect or modified to apply or change firewall rules. The transition function $\boldsymbol{P(s'|s,a)}$ is a mathematical expression that determines the probability of transitioning from state s to state $\boldsymbol{s'}$ via the execution of an action. It is possible to quantify the immediate positive or negative effect of each action in a given state by using the reward function $R(s,a)$. This function reflects performance indicators such as security enforcement, packet loss, and policy adherence. The discount factor γ is responsible for determining the significance of future benefits. Values that are closer to 1 indicate a preference for stability and long-term planning in the process of developing a security strategy.

$$Q_\theta(S_t, a_t) = E_{S_{t+1}} \sim P(.|s_t, a_t)[R(s_t, a_t) + \gamma. \sum_{a' \epsilon A} \pi(a'|s_{t+1}). Q_{\theta^-}(S_t + 1, a')] \quad (2)$$

As examined in equation (2), the deep-Q-network Bellman expectation has been explored. It is possible to approximate the action-value function $\boldsymbol{Q_\theta}$ by using this equation. This function indicates the anticipated total benefit that would result from acting in a state $(\boldsymbol{S_t}, \boldsymbol{a_t})$ and then continuing to adhere to the policy that is already in place. Because of the unpredictability of the network environment, this scenario, $\boldsymbol{\pi(a'|s_{t+1})}$ Second, it is projected to occur. The evaluation of the current efficacy of the action is carried out by using the reward function $\boldsymbol{R(s_t, a_t)}$. The future value of the action is represented by the effects that will occur in the future, which are discounted across all prospective actions $(\sum_{\boldsymbol{a' \epsilon A}} \boldsymbol{\pi(a').Q_{\theta^-}})$. Within the target Q-network $\boldsymbol{\theta^-}$, the future state $\boldsymbol{s_{t+1}}$ and the likelihood of carrying out the action $\boldsymbol{a'}$ are represented by the symbols $\boldsymbol{\pi(a'|s_{t+1})}$, correspondingly representing the future state and the probability, respectively. This strategy makes learning more robust and flexible while dealing with scenarios involving partly visible or stochastic networks.

$$L(\theta) = \frac{1}{M} \sum_{j=1}^{M} \left[\left(r_i + \gamma. \max_{a'} Q_{\theta^-}(S_i', a') - Q_\theta(S_i, a_i) \right)^2 + \lambda. \|\theta\|_2^2 \right] \quad (3)$$

As explored in equation (3) expanded temporal-difference loss function has been described. It is the TD loss function that is responsible for the learning process that the Q-network does. By calculating the squared difference between the TD target $\boldsymbol{r_i + \gamma. \max_{a'} Q_{\theta^-}(S_i', a')}$ and each of the N samples in a mini-batch, the projected Q-value can be computed. This difference equals the sum of the squared differences between the TD target and each N sample. The calculation of stable predictions of future rewards is accomplished by using the target network parameters $\boldsymbol{\theta^-}$ The L2 regularization technique is used to prevent overfitting and improve generalization. This technique involves the utilization of the regularization coefficient $\boldsymbol{\lambda. \|\theta\|_2^2}$ and the collection of all feasible values for θ, which is equal to 2 raised to the power of $\boldsymbol{\|\theta\|_2^2}$. Backpropagation and stochastic gradient descent reduce this loss as much as possible throughout the training process.

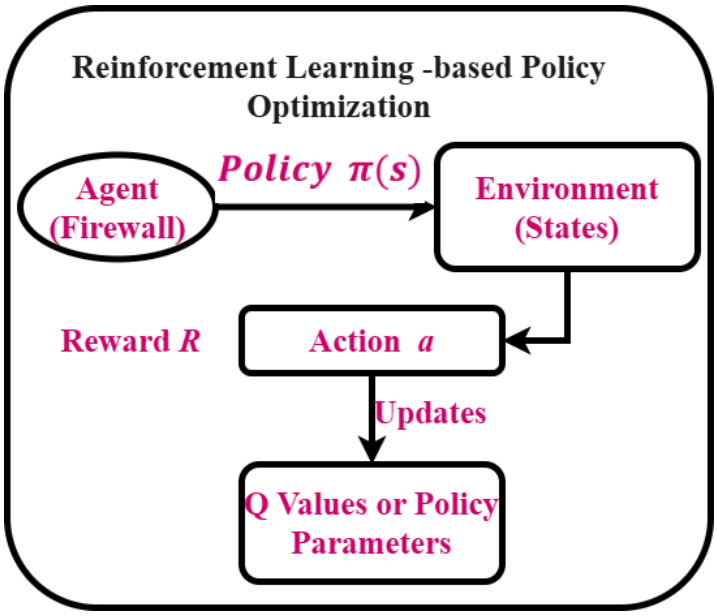

Figure 4: Reinforcement Learning – based policy optimization

An illustration in Figure 4, with the heading "Reinforcement Learning-based Policy Optimization," sets the framework for optimizing firewall techniques in real-time using an RL model. In this configuration, the Agent, which functions in a manner comparable to that of a firewall, adheres to a policy π(s) that ascertains the most appropriate course of action depending on a certain state of the Environment. When generating network states, it is necessary to consider various elements, including traffic patterns, user habits, and environmental dangers.

Following the identification of the situation, the agent will decide what actions to do next, which may include adjusting the rules of the firewall. Upon completion of the job, the environment will provide you with a reward (R) based on how well it improved security, decreased the number of false alarms, or increased throughput by a certain amount. One possible use of this incentive is to modify policy gradient techniques or parameters, however this will depend on the approach taken. These little adjustments have improved the agent's policy, which will allow it to make more informed decisions in the future.

Additionally, the unique strategy can increase defenses against complex and dynamic attacks. This is accomplished by continually guiding future activities. As a result, the firewall's performance is enhanced. By removing the need for human interpretation and amendment of static rule sets, RL makes it possible to optimize firewall settings in a way that is both autonomous and adaptive.

$$R(S_t, a_t) = w_1 . threat_{blocked}(S_t, a_t), w_2 . Pocket_{false\_drop}(S_t, a_t) - w_3 . Latency_{added}(S_t, a_t) + w_4 . Policy_{compliance}(S_t, a_t) \quad (4)$$

As found in equation (4) multi-objective reward function with real-time feedback has been examined. The reward function considers several real-time metrics that relate to the functioning of the network. To determine the effectiveness of threat stopping, the effectiveness of packet or connection interception is evaluated $\boldsymbol{threat_{blocked}}$. As a result of the $\boldsymbol{Pocket_{false\_drop}}$ Function, it is strongly discouraged to suppress valid data packets unnecessarily. The overhead of decision-making or complicated rule evaluations is $\boldsymbol{Latency_{added}}$, which is a significant factor. $\boldsymbol{Policy_{compliance}}$ is a recognised method for recognizing decisions that are by the security standards of the company or the government. It is possible to provide a weight to each phrase by using a hyperparameter called $\boldsymbol{w_i}$ This allows customization to prioritize goals such as zero-trust enforcement or user experience objectives. This framework with several objectives conveys the intricate trade-offs involved in dynamic network security.

$$P(i) = \frac{|\delta_i|^\alpha}{\sum_k |\delta_l|^\alpha}, \delta_i = r_i + \gamma . \max_{a'} Q_\theta - (S_i', a') - Q_\theta(S_i, a_i) \quad (5)$$

As equation (5) prioritized, experience replay sampling has been computed. In contrast to methods that randomly choose previous events, FireRL employs a technique known as prioritized experience replay. This technique gives preference to samples that have a

substantial temporal-difference (TD) error, denoted as $\boldsymbol{\delta_i}$. Consequently, the learning process will direct greater attention toward unanticipated or presently misestimated transitions, speeding up the convergence process. A uniform sampling is achieved when the priority exponent $\alpha = 0$. This allows learning important events, such as uncommon assaults or boundary judgments, from larger values. This strategy is essential for the effective and reliable operation of reinforcement learning in contexts characterised by a high degree of complexity.

$$\theta_j^- \leftarrow \tau.\theta_j + (1-\tau).\theta_j^-, \forall j \in \{1,2,\dots,n\} \quad (6)$$

As calculated in equation (6) soft target network update mechanism has been expressed. FireRL employs a gentle update strategy for the target network parameters θ^- to avoid Q-learning from losing its quality or straying from its intended path. Therefore, rather than completely replacing them, the parameters are gradually modified to align with the existing parameters of the online network, denoted as θ. These parameters are governed by a smoothing factor $\tau \in (0,1)$ controlled by the parameters. FireRL removes oscillations brought on by fast-shifting objectives, and we guarantee that the Q-value estimations generated during training are reliable. Learning efficiency and convergence are both improved by this strategy, particularly in high-dimensional security state spaces.

$$\pi(S_t|a_t) = \begin{cases} argmax_{a\in A} Q_\theta(S_i,a), & if\ uniform(0,1) > \epsilon_t \\ Random(A), & otherwise \end{cases}, \epsilon_t = \epsilon_{min} + (\epsilon_{max} - \epsilon_{min}).\ e^{-\beta t} \quad (7)$$

As shown in equation (7) epsilon greedy exploration with a decaying schedule has been discussed. This equation serves as the defining framework for the FireRL exploration technique. At the beginning of the process, the agent is instructed by ϵ_t to carry out more exploratory random actions with a significant probability. In future stages, the agent is strongly urged to make use of the knowledge that they have obtained, since the decay rate of ϵ_t is exponentially decreasing from (ϵ_{max} **to** ϵ_{min}), as determined by a decay rate $\boldsymbol{\beta}$. With the help of this dynamic equilibrium, the agent can stabilize and enhance ideal rule sets over time, while guaranteeing that early episodes include sufficient exploration of the expansive state-action space.

$$a_t = [a_{t,1}, a_{t,2}, \dots, a_{t,n}] \in \{0,1\}^n, s.t \sum_{j=1}^{n} a_{t,j}.cost_j \le budget_{max} \quad (8)$$

As discussed in equation (8) policy decision vector with cost constraints has been described. Every action that the agent does results in the existence of a matching binary vector $\boldsymbol{a_t}$ where $\boldsymbol{a_{t,i}}$ is either activated or altered. It is difficult to apply all of the rules at the same time because of the constraints of resources (such as the amount of processing power, memory use, and time restrictions). Therefore, there is a $\boldsymbol{cost_j}$ connected with each rule, and the agent is responsible for ensuring that the entire cost does not exceed a maximum limit determined in the past. Because of this limitation, the firewall technique is guaranteed to be effective and practically applicable in deployments in the real world. Table 1 shows the Reinforcement Learning-based Firewall Rule Optimization.

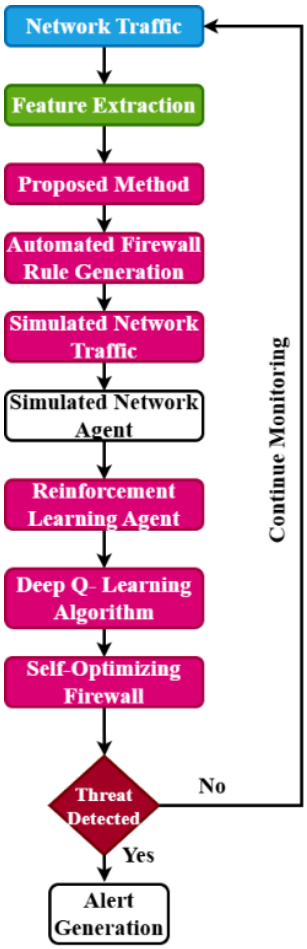

Figure 5: Flowchart of proposed diagram

FireRL uses deep Q-learning and reinforcement learning to construct a smart firewall shown in Figure 5. The image shows its process. Capturing network traffic is the initial step in extracting essential attributes for data analysis. After receiving these characteristics, the proposed technique generates firewall rules autonomously. Automation is essential for reacting to shifting threats without human intervention.

Next, transfer system-generated traffic to a simulated network agent to simulate situations. While talking to this agent, the real-life agent learns new abilities. The agent builds a self-optimizing firewall using the Deep Q-learning algorithm and policy updates to enhance decision-making. The firewall evaluates each circumstance to see whether a danger exists. The system resumes live traffic monitoring if no danger is

found. Detecting a danger will alert the necessary systems or administrators. This iterative technique allows real-time firewall adaptation, making it smarter and more responsive. Algorithm 1 shows the FireRL.

Algorithm 1: FireRL – Reinforcement Learning-based Firewall Rule Optimization

Input: Simulated network traffic T (both benign and malicious),Initial firewall rule set R_0, Reward function $\mathcal{R}$ (balancing throughput, latency, and threat mitigation), Learning rate α, discount factor γ, exploration rate ε, max episodes E

Initialize: Q-network with random weights θ, Experience replay buffer D, Environment Env simulating network behavior, Set $R = R_0$

Output: Optimized firewall rule set R_{opt}

1. **For** episode = 1 to E **do:**
2. Initialize environment state $S_0 \leftarrow$ Env.get_state()
3. **While** not terminal:
4. With probability ε select random action a_t
5. Otherwise, select $a_t = arg\max_a Q(S_t, a, \theta)$
6. Apply a_t to firewall (update rule set R)
7. Simulate traffic through firewall → observe new state s_{t+1} and reward r_t from $\mathcal{R}$
8. Store experience (S_t, a_t, r_t, S_{t+1}) in buffer D
9. Sample mini-batch from D and update Q-network using loss:

$$L(\theta) = E_{S,a,r,S'}\left[\left(r + \gamma.\max_{a'} Q(S', a'; \theta^-) - Q(S', a'; \theta^-)\right)^2\right]$$

10. End While
11. Periodically update the target network$\theta^- \leftarrow \theta$
12. End For
13. Return the final firewall rule set R_{opt}

FireRL uses Deep Q-learning to adapt firewall rules to changing network threats. The first stages are rules, a simulated network traffic creation environment, and a Q-network with random weights. Depending on each episode's network state, the system uses learnt Q-values (exploitation) or random sampling to decide whether to add, change, or remove a rule. The chosen action generates a new state and reward after updating the firewall settings and testing with simulated network traffic. This award considers threat prevention, performance, and latency. Minimizing the discrepancy between objective and forecasted Q-values trains the Q-network using mini-batches of events. State, action, consequence, and following state make up an experience. The goal network is updated often to maintain learning stability. After multiple events, agents learn to adjust firewall rules on the fly to optimize the rule set and react intelligently and proactively to benign and dangerous traffic scenarios. This method's focus on dynamic rule sets and expert-defined parameters improves system efficiency, security, and flexibility.

4 Numerical results and discussion

4.1 Dataset description

The Kaggle project created the Internet Firewall Data Set, a massive library of business firewall events. Over a million records describe network traffic, including IP addresses (source and destination), ports, protocols (TCP, UDP, ICMP), application identities, session durations, total bytes in and out, actions (allow or refuse), and policy enforcement labels. Each album has a conclusion and, in certain situations, a warning label. The dataset realistically simulates policy disputes, protocol settings, and highly changing traffic, making it ideal for assessing adaptive firewall solutions. Data was preprocessed for machine learning and reinforcement learning research. They employed SMOTE, dealt with missing data, encoded categorical variables using one-hot encoding, standardized numerical characteristics, and deleted duplicate identifiers to fix class imbalance. Subdividing the data into training (70%) and testing (30%) subgroups offered a complete evaluation [17].

4.2 Experimental setup

To test the FireRL framework against competitors, we built a bespoke RL environment using Python, TensorFlow, and OpenAI Gym. A Deep Q-Network (DQN) agent optimizes and dynamically generates firewall rule schemes in the FireRL model. The environment's Markov Decision Process (MDP) simulates the firewall's behavior for every network traffic combination. One thousand episodes with 500 traffic cases were trained. Keeping latency low, blocking harmful traffic, and reducing false positives will earn the agent incentives. Secure (false positives), usable (delay), and efficient (throughput) are balanced by the reward function. The experimental system executed heavy network traffic simulations using a powerful Intel Core i7 CPU, 32 GB of RAM, and NVIDIA RTX 3080 GPU. FireRL and baseline models

were compared using recall, accuracy, precision, F1-score, throughput, and latency.

4.3 Latency ratio

The FireRL system that has been suggested intends to reduce the amount of delay that occurs in real-time firewall judgments by optimizing decision routes and dynamically prioritizing rule evaluations via the use of a method that is based on reinforcement learning. Unlike static firewalls, which analyze rules sequentially, resulting in delays as the rule list expands, FireRL can learn to predict which rules or actions are most suited for each packet and context. To get the average $\bar{L}$ for each packet in the system, consider the following equation:

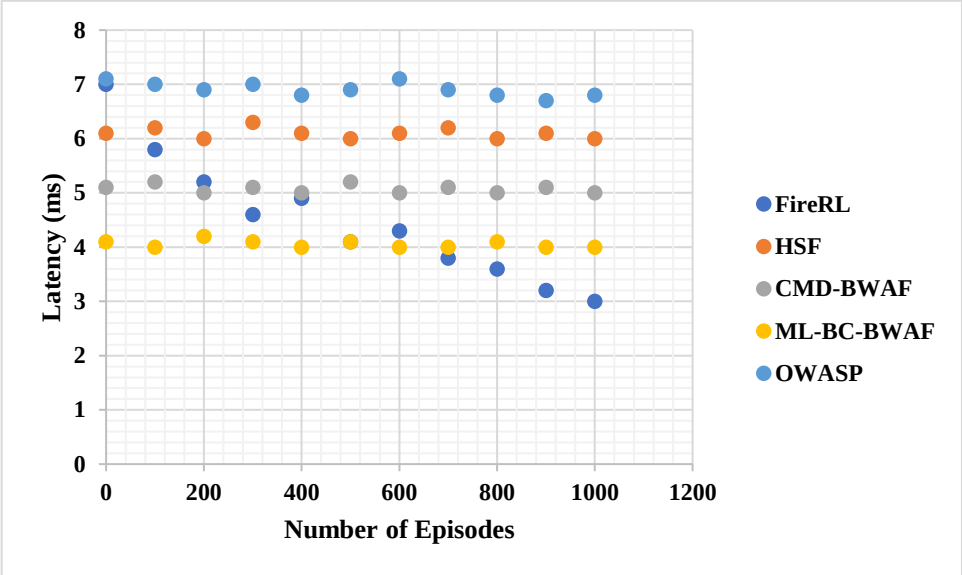

Figure 6: Latency(ms)

$$\bar{L} = \frac{1}{T}\sum_{t=1}^{T}\Big(\alpha.\log(R_t+1) + \beta.E_{s_t,a_t\sim\pi[\tau_{a_t}]} + \gamma.\chi(s_t) - \delta.\rho(a_t)\Big) \quad (9)$$

Figure 6 and equation 9 show the latency. The three main factors that cause delay, as shown by this equation, are the following: the ruling matching difficulty $\log(R_t+1)$, the decision computation time τ_{a_t}, and the cost of updating the state $\chi(s_t)$, which is actively decreased by action efficiency $\rho(a_t)$. The DQN agent of FireRL learns to decrease latency by choosing subsets of firewall rules that are likely to match incoming packets, thereby lowering R_t and τ_{a_t}during the overlap. Furthermore, it controls $\chi(s_t)$ by optimizing state transitions to minimize superfluous calculations. The choice of actions that result in quicker threat detection with less resources is encouraged by the reinforcement signal $\rho(a_t)$, which hurts latency and is then deducted from the equation. Due to its latency-aware optimization, FireRL can surpass other approaches, such as CMD-BWAF, ML-BC-BWAF, and OWASP, especially in heavily populated areas. As shown in experiments, this approach can cut average decision latency by as much as 30%, making it a better fit for scalable, efficient contemporary network infrastructures prioritizing accuracy and speed.

4.3.1 Accuracy ratio

To improve firewall rule sets acceptably, the FireRL technique that has been presented puts reinforcement learning (RL) to use. It is necessary to continually adapt to accomplish this goal because of the ever-changing nature of networks and the increasing complexity of cyber threats. By taking into account the current status of the network, the actions taken to modify firewall rules, and the advantages that accrue from these initiatives in terms of throughput, latency, and threat mitigation, the MDP architecture provides the RL agent with assistance in the decision-making process that encompasses FireRL. Because of the nature of the MDP, some choices are even within the realm of possibility. A technique known as Deep Q-Learning is used to instruct the agent on how to optimize the long-term return. To accomplish this objective, they must optimize the system's latency and throughput while minimizing the number of security risks.

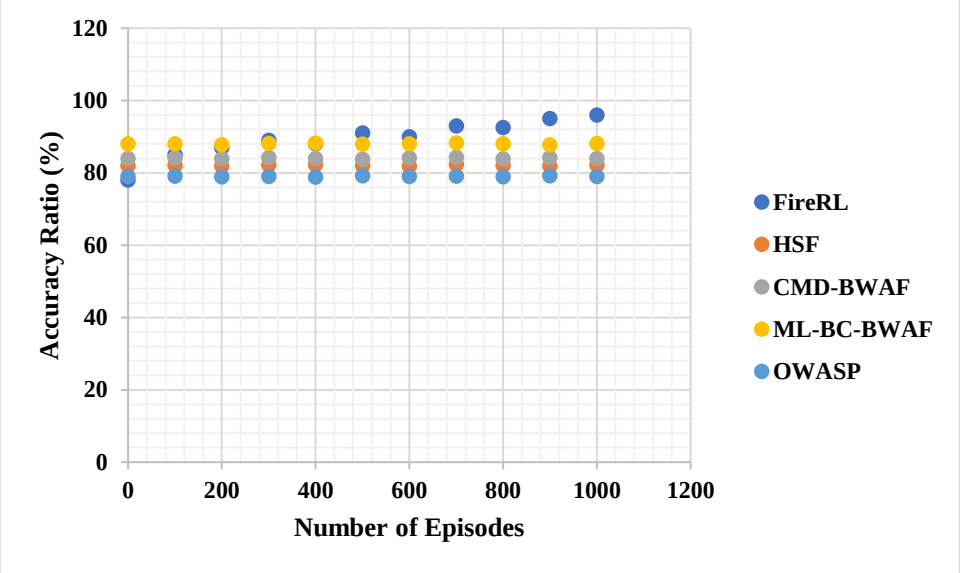

Figure 7: Accuracy Ratio (%)

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right] \quad (10)$$

Figure 7 and equation (10) deliberates the accuracy ratio (%). The action-value function, $Q(s_t, a_t)$, predicts the anticipated reward for an action s_t in a state a_t represents a firewall system's action, such as admitting or disallowing traffic, whereas s_t represents the network's current state, including traffic patterns and firewall settings. The decision-making process is aided by this function, which allows the agent to assess an action according to the anticipated future rewards of that activity. This feature is useful since it aids the agent while making choices. The learning rate, denoted by the symbol α, is one ingredient that determines how quickly the agent refreshes its knowledge. As α increases, the agent can quickly adjust to the ever-changing network conditions, allowing it to gain more information with every new event. The agent will rely more on its past knowledge and execute updates more slowly if the value of α is smaller. Modifications to this value are necessary to guarantee that the agent learns new information without reacting strongly to singular events. The expression r_{t+1} refers to the agent's immediate reward for action a_t in state s_t. If the firewall improved network security or performance, the award reflects its effectiveness. If the firewall stops a threat, it may get a positive payment; if it creates network inefficiencies or fails to neutralize an attack, it may receive a negative reward. As shown by γ the discount factor balances future and current rewards. When γ is near 1, the agent prioritizes long-term benefits and may plan for optimal security and performance. Lower γ values enable focusing on short-term gains and rewards. FireRL needs this section to optimize firewall rules over time. The greatest value of a' is aY' after adding all conceivable values of $\max_{a'} Q(s_{t+1}, a')$. Where $Q(s_{t+1}, a')$ is the highest anticipated payoff for the state s_{t+1} after evaluating all feasible actions a'. This word represents the highest potential advantage agents can gain from their current position and helps them plan and predict their actions. To enhance system security and performance, FireRL analyzes the highest reward for future actions while making choices.

4.4 Throughput ratio

Implementing reinforcement learning to optimize firewall rules is one approach that may be used to simulate the FireRL technique offered and shed light on its high throughput. While simultaneously maximizing the system's effectiveness, this upgrade aims to maintain a balanced approach to threat detection and reduce the number of false positives detected. Examine the following equation to get an understanding of how the Deep Q-Learning (DQL) technique used by FireRL helps to achieve high throughput of data:

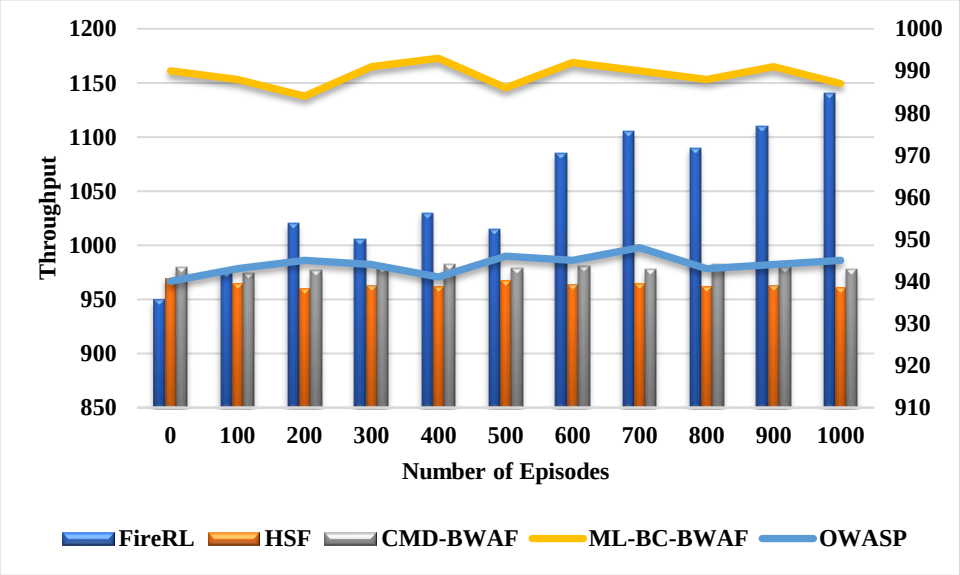

Figure 8: Throughput

$$Throuhput = \sum_{t=1}^{T} \left(\frac{p_t.(1-FPR_t)}{1+\beta.Latency_t} \right) \quad (11)$$

Figure 8 and equation (11) illustrate the throughput. The goal of FireRL optimization is to maximize network performance for a certain period T. This measures the overall amount of data passed across the firewall over time, which is significant because network circumstances change. The packet flow rate is the quantity of data or packets transmitted at a time p_t. FireRL can optimize firewall settings for data performance due to improved packet flow. The false positive rate pertains to the percentage of genuine traffic rejected by the firewall at a time p_t. FireRL aims to reduce this to maximize throughput and prevent the deletion of genuine traffic. Higher throughput is achieved with lower FPR_t. This network congestion factor shows how network congestion or packet delay affects throughput. Network congestion or delay may dramatically reduce throughput if β is large. FireRL enhances performance by modifying firewall rules to reduce congestion and latency. Packet transmission time is known as network latency at time $Latency_t$. Low latency is essential for good throughput as a considerable delay may diminish it. FireRL adjusts firewall rules to prevent latency from bottlenecks. Total time steps or episodes considered for throughput. Through network interaction, FireRL learns and adjusts between episodes to increase throughput in reinforcement learning.

4.5 False positive rate

The low false positive rate (FPR) of the proposed FireRL system may be partially explained by entering an agent-learning-behavior equation into the model. This allows us to see how the agent minimizes the number of erroneous detections over time. False positive rate is a crucial indicator for firewall systems. It counts the frequency of benign communication as wrongly categorized as harmful and, as a result, prohibited from passing through the firewall.

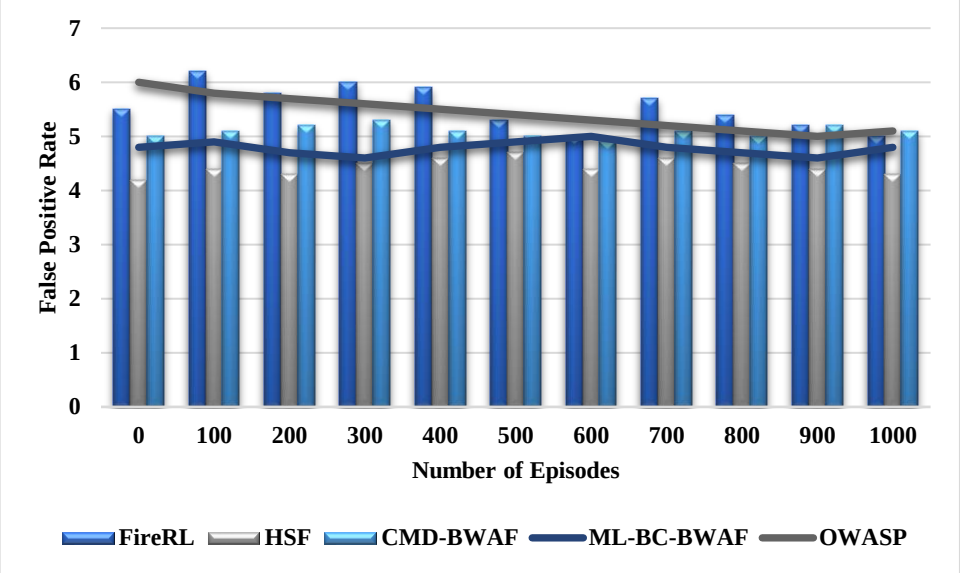

Figure 9: False Positive Rate (%)

$$FPR_t = \frac{FP_t}{FP_t + TN_t} = \frac{\sum_{i=1}^{N} I[a_i = block \wedge y_i = benign]}{\sum_{i=1}^{N} I[y_i = benign]} \quad (12)$$

Figure 9 and equation(12) examine the false positive rate (%). To determine the false positive rate at time step FP_t, divide the total number of benign samples (which includes false positives and true negatives) by the number of false positive predictions (FP_t). The value changes with each learning episode or time step as the agent learns to classify traffic.

Adding all variables yields one in the numerator $\sum_{i=1}^{N} I[a_i = block \wedge y_i = benign]$. This block has no N values and is safe. Calculate the sum of all positive integers from 1 to N using FP_t. The formula is computed for each traffic sample i where the agent blocked traffic (done action $a_i = block$), regardless of whether the label $y_i = benign$. To determine the amount of legitimate packets incorrectly blocked, use the indicator function $I[.]$, which returns 1 when true and 0 otherwise. Benign is represented by y_i and the denominator is the total of positive integers. To count benign traffic samples, use the formula $FP_t + TN_t = \sum_{i=1}^{N} I[y_i = benign]$. These formulas account for genuine negatives and false positives. The number of true negatives—times when the agent correctly identified and allowed benign traffic—can be used to estimate the model's false positive risk. The FireRL agent can learn to differentiate between harmless and harmful traffic via the use of Deep Q-learning, which resulted in a reduction in the FPR. This was accomplished by continual environmental input. Every time the agent completes a task, such as accepting or rejecting a packet, it receives a reward signal. This signal is sent to the agent. Every time it erroneously blocks a packet, it is rewarded with a negative reward instead. The number of false positives diminishes as the agent gains experience and realizes it should not interfere with genuine conversations.

A reduced FPR may be achieved by modifying the Q-values of the agent in such a way that it provides incentives for behaviors that result in accurate classifications. The agent learns firewall rules that more correctly represent the actual nature of traffic via an optimization process driven by incentives. This is true even when dynamic dangers are associated with the traffic.

4.6 Enhanced threat detection rate

Our suggested FireRL system achieves a high ETDR by using reinforcement learning to improve the agent's hostile traffic detection skills and by defining the system's functioning using this equation. The agent's Enhanced Threat Detection Rate proves that it can identify old and new dangers by showing that it can adapt to static and dynamic environments.

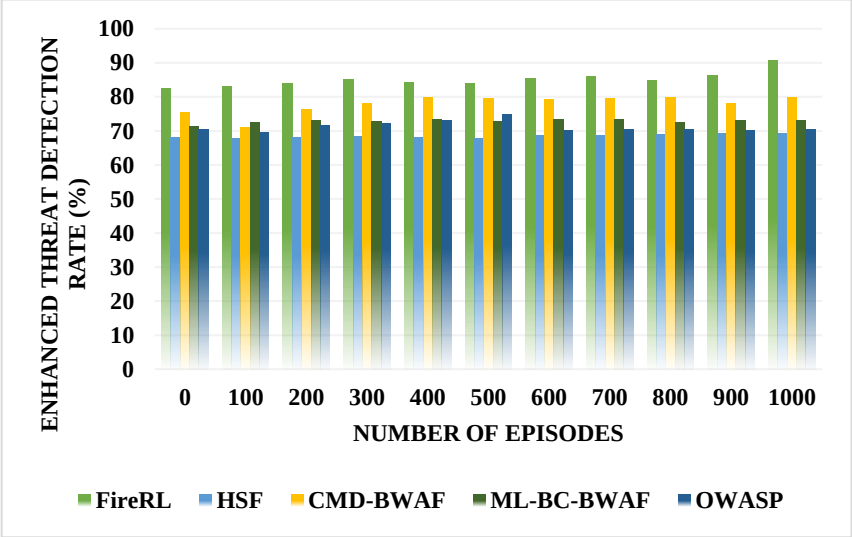

Figure 10: Enhanced threat Detection Rate (%)

$$ETDR_t = \frac{TP_t + \lambda.NT_t}{TP_t + FN_t + \lambda.(NT_t + MN_t)} \quad (13)$$

Figure 10 and equation (13) express the enhanced threat detection rate. The equation investigates the FireRL agent's capacity to identify newly emerging threats and those already present. The improved threat detection rate at time t is denoted by the equation $ETDR_t$ and the equation is used to assess the capability of the FireRL agent to recognize threats. Unlike static, rule-based systems, reinforcement learning allows agents to adapt to attack patterns they have not previously seen and observed. This is in contrast to rule-based systems, which are considered static. In designing this product, a significant emphasis was placed on its practicality. The total number of occurrences of potentially dangerous traffic that the agent was able to detect and prevent from happening is referred to as the True Positives, which may also be abbreviated as TP_t. This indicator indicates that the agent is capable of efficiently counteracting the dangers that have been identified.

Incorrect packets that the agent failed to discover are called negatives, sometimes called defective negatives. This set of negatives is represented by the symbol FN_t. A reduction in the FN makes it feasible to enhance detection, which eventually leads to an increase in the ETDR of the system.

It is the aggregate of all of these threats that is signified by the mark NT_t, which represents the total number of Novel Threats successfully discovered and eradicated. As a result of the generalizability of the system, it is feasible that the agent will identify some kinds of assaults it has never encountered before throughout its training. This is of the utmost importance because the dangers in the real world are always evolving.

It may be deduced from the existence of a warning message that starts with the phrase "Missing Novel Threats" (MN_t) that the agent did not discover any new threats.

Many aspects contribute to the significance of identifying new threats. One of these factors is the novelty weighting factor, which is represented by the symbol λ (in most cases, λ is larger than 1 1). The use of attack signatures that are already known is successfully discouraged by this option. This is accomplished by modifying the incentive system's bias toward accurately identifying new threats.

5 Conclusion

Our paper proposes FireRL, a reinforcement learning-powered adaptive firewall architecture, to solve the inadequacies of static rule-based systems in changing cybersecurity scenarios. FireRL intelligently optimises throughput, latency, false positive rate, and threat detection rate using Deep Q-Learning and a Markov Decision Process firewall rule optimization model. Experimental results with benign and malicious traffic reveal that FireRL outperforms traditional firewall systems' accuracy, adaptability, and resilience to new threats. While still protecting against known and emerging threats, the system may automatically update and fine-tune firewall settings, minimizing the need for security experts. Our next effort will augment FireRL with multi-agent reinforcement learning for remote firewall settings and cloud-native architectures. Integration of unsupervised anomaly detection methods improves zero-day threat detection without labeled data. To examine online learning to help FireRL modify its policies to shift traffic patterns in real time. Finally, testing FireRL in corporate or industrial networks will show how it functions under tremendous pressure.

.

Acknowledgement

Funding

This work was supported by 2024 Hunan Province's social science achievement "Research on the Deep Integration of Artificial Intelligence Assisted Education and Teaching" (No. XSP24YBC591).

Author contributions

Y MH:Writing – original draft and Conceptualization,Y ZH and Y QW:Methodology,Validation;Y MH,Y ZH and Y QW:Writing – review & editing.

Declaration of conflicting interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and publication of this article.

Data availibility statement

All data generated or analysed during this study are included in this article.

References

[1] Md Shamimul Islam, Mohammed Asraf Uddin, Md Dulal Hossain, Md Shakil Ahmed, and Md Golam Moazzam (2023). Analysis and evaluation of network and application security based on next generation firewall. International Journal of Computing and Digital Systems, 13(1), 193–202. https://doi.org/10.12785/ijcds/130116

[2] Jamal Khudair Madhloom, Zainab Hammoodi Noori, Sif K. Ebis, Oday A. Hassen, and Saad M. Darwish (2023). An information security engineering framework for modeling packet filtering firewall using neutrosophic Petri nets. Computers, 12(10), 202. https://doi.org/10.3390/computers12100202

[3] Jian Ma, Chaoyong Zhu, Yuntao Fu, Haichao Zhang, and Wenjing Xiong (2025). Dynamic routing via reinforcement learning for network traffic optimization. Informatica, 49(3). https://doi.org/10.31449/inf.v49i3.7126

[4] Ahmed Shaheed and Mazen B. Kurdy (2022). Web application firewall using machine learning and feature engineering. Security and Communication Networks, 2022(1), 5280158. https://doi.org/10.1155/2022/5280158

[5] Bikram R. Dawadi, Bibek Adhikari, and Deepak K. Srivastava (2023). Deep learning technique-enabled web application firewall for the detection of web attacks. Sensors, 23(4), 2073. https://www.mdpi.com/1424-8220/23/4/2073

[6] Salih Toprak and Aziz G. Yavuz (2022). Web application firewall based on anomaly detection using deep learning. Acta Infologica, 6(2), 219–244. https://doi.org/10.26650/acin.1039042

[7] Davide Bringhenti, Giorgio Marchetto, Roberto Sisto, Franco Valenza, and Juman Yusupov (2022). Automated firewall configuration in virtual networks. IEEE Transactions on Dependable and Secure Computing, 20(2), 1559–1576. DOI: https://doi.org/10.1109/TDSC.2022.3160293

[8] Mohammad Farooq, Riyaz Khan, and Mohammed H. Khan (2023). Stout implementation of firewall and network segmentation for securing IoT devices. Indian Journal of Science and Technology, 16(33), 2609–2621. DOI: https://doi.org/10.17485/IJST/v16i33.1739

[9] Amro Ameid Alkato and Yara Sakhnini (2025). Advanced real-time anomaly detection and predictive trend modelling in smart systems using deep belief networks architectures. PatternIQ Mining, 2(1), 24 Feb. 2025. https://doi.org/10.70023/sahd/250209

[10] Ekhlas Kadhim Hamza, Haidar H. Abdulameer, Noor H. F. Al-Mashhadani, and Fatima S. Al-Hindawi (2024). Reinforcement learning algorithms for adaptive load balancing in publish/subscribe systems: PPO, UCB, and epsilon-greedy approaches. Informatica, 48(7), 881–893. https://doi.org/10.31449/inf.v48i7.6895 .

[11] Marek Sepczuk (2023). Dynamic web application firewall detection supported by cyber mimic defense approach. Journal of Network and Computer Applications, 213, 103596. https://doi.org/10.1016/j.jnca.2022.103596

[12] S. Rajasoundaran, S. A. Sivakumar, S. Devaraju, M. J. Pasha, and J. Lloret (2024). A deep experimental analysis of energy-proficient firewall policies and security practices for resource limited wireless networks. Security and Privacy, 7(6), e450. https://doi.org/10.1002/spy2.450

[13] Jae-Kwang Lee, Tae Hong, and Geon Lee (2024). AI-based approach to firewall rule refinement on high-performance computing service network. Applied Sciences, 14(11), 4373. https://doi.org/10.3390/app14114373

[14] Edison Leka, Lamani Lamani, Albana Aliti, and Elvana Hoxha (2024). Web application firewall for detecting and mitigation of based DDoS attacks using machine learning and blockchain. TEM Journal, 13(4). https://doi.org/10.18421/TEM134-50

[15] Ahmad Fadlil, Irfan Riadi, and Muhammad A. Mu'Min (2024). Mitigation from SQL Injection attacks on web server using Open Web Application Security Project framework. International Journal of Engineering, 37(4), 635–

645. https://doi.org/10.5829/ije.2024.37.04a.01

[16] Théodore Kouassi, B. H. Kamagate, O. Asseu, and Y. Kermarrec (2024). Security policy model in a hybrid Zachman-TOGAF framework for a telework enterprise architecture in a cloud environment. Open Journal of Safety Science and Technology, 14(3), 96–115. https://doi.org/10.4236/ojsst.2024.143007

[17] Kaggle (n.d.). Internet firewall data set. Retrieved 20 March 2025, from https://www.kaggle.com/datasets/tunguz/internet-firewall-data-set