Beyond Reinforcement Learning for Network Security: A Comprehensive Survey and Tutorial

Amir Javadpour, Forough Ja'fari, Tarik Taleb, Fatih Turkmen and Chafika Benzaïd

***Abstract*—Maintaining strong security is a complex yet vital challenge in the rapidly evolving landscape of modern digital networks. The risks and consequences of security breaches make neglecting network protection unacceptable. Fortunately, ongoing advances in computer science have equipped researchers with powerful tools to reinforce network defenses. Among these, Reinforcement Learning (RL), a branch of machine learning, has gained significant attention for its versatility and effectiveness in strengthening security mechanisms. This paper presents a comprehensive survey and tutorial on the role of RL in network security. It provides background information, a step-by-step tutorial for training RL models, and systematically categorizes research efforts based on the targeted cyber threats. Leveraging recent advances and real-world applications, this survey elucidates how RL enables the development of adaptive and intelligent systems that autonomously learn and respond to evolving threats. Through in-depth analysis, we provide a comprehensive view of the current landscape and the future potential of RL in safeguarding digital assets. The main contributions of this survey are: (1) a systematic and up-to-date review of RL approaches for network security; (2) a unified taxonomy for classifying RL-based solutions; (3) a comparison of the latest advances from 2019 to 2024 across mainstream and emerging research areas; (4) identification of open challenges and future research directions; and (5) a comparative analysis of state-of-the-art models, offering practical insights for both researchers and practitioners. Furthermore, this survey emphasizes the practical translation of RL advances into real-world deployments. By focusing on hands-on implementation guidelines and comparative analyses of deployment scenarios, it bridges the gap between academic research and operational practice. The comprehensive evaluation of RL-based models across different network environments provides actionable insights for practitioners seeking adaptive and scalable security solutions in dynamic and heterogeneous settings.**

***Index Terms*—Network Security; Comprehensive Survey and Tutorial Security; Reinforcement Learning for Network Security; Attack Mitigation**

CONTENTS

Amir Javadpour (Senior Cybersecurity Researcher MOSA!C Lab / ICT-FICIAL Oy, Finland),
Forough Ja'fari (Sharif university of technology),
Tarik Taleb (Faculty of Electrical Engineering and Information Technology, Ruhr University Bochum, Bochum),
Fatih Turkmen (University of Groningen, The Netherlands)
Chafika Benzaïd (Faculty of Information Technology and Electrical Engineering, University of Oulu)
This research is partially supported by the European Union's Horizon Europe research and innovation program under the **RIGOUROUS** project (Grant No. 101095933). It is also partially supported by the Research Council of Finland (formerly the Academy of Finland) through the 6G Flagship program (Grant No. 346208). The paper reflects only the authors' views, and the European Commission bears no responsibility for any utilization of the information contained herein.
Corresponding author: Amir Javadpour (a.javadpour87@gmail.com)

I. INTRODUCTION

THE prevalence of machine learning in our daily lives is on a swift ascent, impacting various facets of our daily routines. Among these, ensuring the security of our digital interactions, particularly over the internet, has become an indispensable requirement that often operates behind the scenes, unnoticed by many. In an era where our personal and professional lives are increasingly intertwined with digital networks, safeguarding the privacy and integrity of the information we transmit has never been more paramount [1, 2]. As individuals, we entrust sensitive data to digital channels for tasks as routine as online shopping, communication, and financial transactions. For organizations, securing these digital pathways is crucial to maintaining trust and compliance. Within the realm of machine learning, RL stands out as a powerful tool that security specialists are increasingly adopting. It plays a pivotal role in fortifying the security of digital networks, which virtually all of us have become a part of, whether knowingly or unknowingly. This technology enables networks to adapt dynamically to emerging threats and vulnerabilities, enhancing our digital experiences and ensuring that our shared data remains protected. In essence, RL has quietly become an indispensable guardian of the digital realm, fortifying the networks that underpin our modern way of life. Evidence of this is the recently established Cage Challenge [3], which aims to foster the development of autonomous cyber defence agents.

The sharp increase in the use of RL for securing digital networks, and the lack of a detailed survey on this topic has motivated us to work on this paper, which is a complex task in this very active area of research and the fast-changing digital world. Neglecting information technology security can lead to serious threats and vulnerabilities. Fortunately, advancements in computer science, particularly in machine learning, have equipped researchers with tools to combat these security issues. There are some survey papers summarizing different uses of RL for securing computer networks and in general cybersecurity, such as those presented by Adawadkar and Kulkarni [4], Luong et al. [5], Nguyen and Reddi [6], and Uprety and Rawat [7]. However, they lack focused details on network security aspects and do not have a step-by-step tutorial through examples for those who want to employ RL in their security related work or become a security specialist/researcher. This paper aims to present an exhaustive and in-depth survey of the utilization of RL in the realm of *network security*. In today's interconnected world, where data and information flow seamlessly across networks, maintaining the integrity, confidentiality, and availability of this digital infrastructure is paramount. Threat actors, ranging from cyber criminals and hacktivists to state-sponsored entities, continually develop new attack vectors and techniques to compromise network security. As a result, protecting against these threats and automation require continuous adaptation and innovation. Reinforcement Learning, as an integral component of the machine learning toolkit, offers unique advantages in addressing the dynamic nature of network security. It enables security professionals to create adaptive and intelligent systems that can autonomously learn and respond to emerging threats. By combining computational power and self-improvement through experience, RL has the potential to revolutionize the way we protect digital assets. The primary objective of this paper is to provide an extensive exploration of the role of RL within the context of network security. We delve into the fundamental concepts and principles of RL and elucidate its relevance and applications in securing computer networks. Additionally, we survey and analyze a wide array of research studies, case studies, and practical implementations that have harnessed RL techniques to bolster network security. Through this comprehensive survey, we aim to offer a holistic view of the current state of RL in network security, shedding light on the challenges, achievements, and prospects of this exciting and evolving field.

This paper provides a tutorial-based comprehensive survey by reviewing the recent papers on using RL in network security. The main questions this survey paper aims to answer are as follows:

- What is the step-by-step guideline to generate an RL model?
  This survey aims to provide readers with a clear, step-by-step guide on creating and implementing RL models in the context of network security. Breaking down the process into manageable steps equips readers with practical insights and knowledge necessary to develop their own RL models.
- Which cyber threats are mitigated with RL models in recent research efforts?
  By reviewing recent research papers, this survey paper seeks to identify and categorize the various cyber threats and vulnerabilities that have been successfully addressed and mitigated using RL models. It provides a comprehensive overview of the threats that RL is particularly adept at countering.
- How can we enhance an RL model for network security and What are the future directions of RL in network security?
  In the constantly evolving field of network security, ongoing improvements and enhancements to RL models are crucial. This survey explores the latest methodologies, techniques, and best practices for enhancing RL models to make them more effective and robust in the realm of network security.

Despite the growing body of research on reinforcement learning for network security, several significant gaps remain. First, recent advances are scattered across multiple subdomains with limited integration, making it challenging for researchers and practitioners to obtain a holistic view of the field. Second, most existing surveys either lack comprehensive coverage of the latest literature or do not offer a unified taxonomy that facilitates comparison across approaches. Third, there is a scarcity of in-depth comparative analyses of RL-based models regarding their effectiveness, scalability, and real-world applicability. These gaps underscore the need for an up-to-date, systematic, and comparative survey that consolidates recent developments, identifies open challenges, and offers practical insights for advancing the state of the art in RL-driven network security.

A. Motivation and Goals

As the digital world expands and cyber threats intensify, the importance of network security cannot be overstated. The need for adaptive, flexible, and intelligent security solutions is becoming increasingly apparent. The motivation of this article is to provide valuable insights for students and researchers interested in the practical applications of RL and artificial intelligence in the field of security. Many researchers in this area focus on utilizing existing tools to enhance security and develop solutions that create safer environments. Additionally, to present relevant examples that demonstrate the application of RL and AI for security enhancements. In addition to enhancing security through the use of artificial intelligence, this research presents a step-by-step approach to applying it from an algorithmic perspective. Researchers can learn to work with the RL process first with this tutorial and then apply it to the scenarios they are considering.

This survey aims to introduce and explore the application of RL in the context of network security. It serves as a comprehensive tutorial for training RL models specifically for network security. Also, the survey reviews existing research on the use of RL in various aspects of network security, with a focus on issues such as DDoS attacks and intrusion detection. Furthermore, it discusses the implementation of RL in network defense and evaluates different methods for assessing their effectiveness.

The main objectives of this survey are as follows:

- To provide a comprehensive and up-to-date review of reinforcement learning (RL) approaches for various network security problems.
- To propose a unified taxonomy for classifying RL-based security solutions.
- To systematically analyze and summarize the most recent advances (2019–2024), including both mainstream and emerging research trends.
- To identify existing research gaps and outline future research directions in the field.
- To perform a comparative analysis of state-of-the-art RL models, highlighting their strengths, limitations, and practical implications.

To make this survey broadly beneficial, we have strived to present foundational reinforcement learning concepts alongside the latest developments in an accessible manner. We aim to bridge the gap between theoretical research and practical deployment, enabling both newcomers to the field and experienced professionals to gain valuable insights. By integrating a practical, hands-on tutorial within the survey, we provide a step-by-step guide that facilitates the implementation of RL techniques for real-world network security challenges.

B. Paper Structure

This paper is organized as follows. A detailed background of RL is presented in Section II, which includes its terminology and the algorithms and equations related to the different phases of training and testing for an RL model. Moreover, a complete example of how to design and train an RL model, as well as the related algorithms, are provided in this section. The recent research proposing a RL model for securing networks is reviewed in Section III and categorised based on the threats it focuses on. The suggestions for improving RL models' performance, scalability, and security are discussed in Section IV, and finally Section VI gives a summary and conclusion of this paper. As shown in Figure 1, the framework integrates tutorial content, taxonomy, and comparative insights in a unified view.

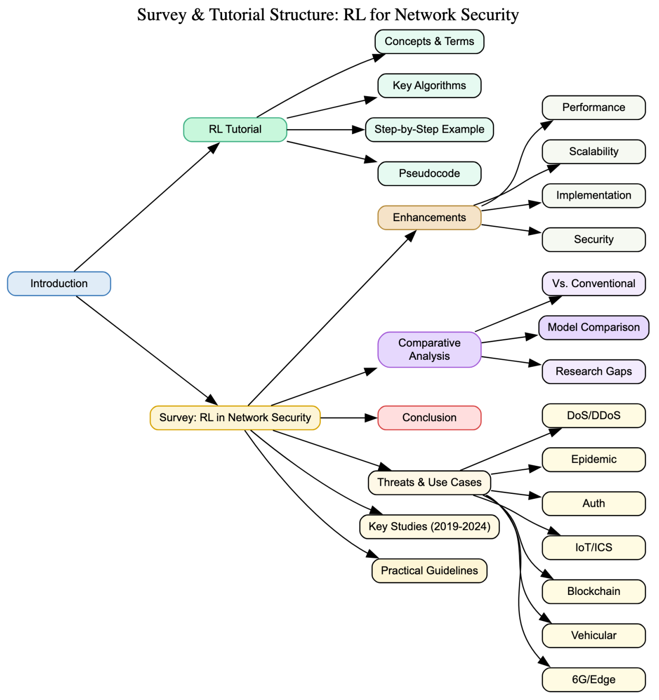


Fig. 1. Overview of the survey and tutorial structure on RL for network security, covering foundational concepts, threat applications, model enhancements, and research gaps.

II. Reinforcement Learning

In this section, an overview of RL and the key terms used in this field are explained. A summary of this terminology is presented in Table I and further explanations for additional variables or terms are provided in the list below as needed. *Note:* All variables and notations (s, a, r, $\mathcal{S}$, $\mathcal{A}$, π, $\mathcal{Q}$, $\mathcal{V}$, γ, α, θ, etc.) are defined in detail in the Notation section above.

Variable and Notation Definitions:

- s: State of the environment.
- a: Action taken by the agent.
- r: Reward received after taking action a in state s.
- $\mathcal{S}$: State space (the set of all possible states).
- $\mathcal{A}$: Action space (the set of all possible actions).
- π: Policy (mapping from states to action probabilities).
- π_θ: Policy parameterized by θ.
- $\mathcal{Q}(s,a)$: Action-value (Q-) function (expected cumulative reward for taking action a in state s).
- $\mathcal{Q}_*(s,a)$: Optimal action-value function.
- γ: Discount factor (importance of future rewards, $0 \leqslant \gamma \leqslant 1$).
- α: Learning rate (step size for updates).
- θ: Policy parameters.

TABLE I
THE ALPHABETIZE REINFORCEMENT LEARNING TERMINOLOGY USED IN THIS PAPER

Term	Notation	Description
Action	a_t	The defined activities the agent can perform in a specific time step, t, which result in state transition.
	a_t^τ	The defined activities the agent has performed in a specific time step, t, in a trajectory, τ.
Action Selection Probability	$\pi(a \mid s)$	The probability of an action, a, to be selected by the agent in a specific state, s.
	$\pi_\theta(a \mid s)$	The probability of an action, a, to be selected by the agent following a specific policy, π_θ, in a specific state (s).
Action Space	$\mathcal{A}$	The set of actions the agent can take, which cause a state transition.
Action Value	$\mathcal{Q}(s,a)$	The cumulative reward the agent can receive doing a specific action, a, in a specific state, s (Equation 6 and Equation 10).
	$\mathcal{Q}_{\pi_\theta}(s,a)$	The cumulative reward the agent can receive doing a specific action, a, in a specific state, s, and following a specific policy, π_θ (Equation 6 and Equation 10).
Optimal Action Value	$\mathcal{Q}_*(s,a)$	The maximum cumulative reward the agent can receive doing a specific action, a, in a specific state, s (Equation 7 and Equation 11).
Discount Factor	γ	A parameter indicating the importance of future rewards over the current reward.
Immediate Reward	$\mathcal{R}(s,a)$	A number that indicates the performance of carrying out a specific action, a, in a specific state, s.
Initial State Probability	$\rho(s)$	The probability of a specific state, s, to be the initial state of the environment.
Learning Rate	α	A parameter indicating the amount of newly learned knowledge that is used for updating the general learned knowledge.
Observation States	$\mathcal{S}$	The set of states, each of which describes the problem environment in its different situations.
Policy	π_θ	The policy, in terms of action probabilities, that the agent follows, which can be described using specific parameters (θ).
Policy Parameters	θ	The parameters that describe the agent's policy.
Q Function	$\mathcal{Q}$	A function that gets a state and an action, and then returns the cumulative reward the agent can receive starting from that state and doing that action as the first action (Equation 6 and Equation 10).
Return Value	$\mathcal{G}_\tau(t)$	The cumulative discounted reward obtained by the agent after a specific time (t) during a trajectory, τ (Equation 3 and Equation 4).
Reward Function	$\mathcal{R}$	A function that gets the current state and the action taken by the agent, and then returns a number that indicates the performance of that action in that state.
State	s_t	The presentation of the problem environment in a specific time step, t.
State in a Trajectory	s_t^τ	The presentation of the problem environment in a specific time step, t, in that trajectory, τ.
State Transition Probability	$\mathcal{P}(s' \mid s,a)$	The probability of transitioning from one state, s to another, s', performing a specific action, a.
State Value	$\mathcal{V}(s)$	The expected cumulative reward the agent can receive starting from a specific state, s.
	$\mathcal{V}_{\pi_\theta}(s)$	The expected cumulative reward the agent can receive starting from a specific state, s, following that policy, π_θ (Equation 5 and Equation 9).
Trajectory	τ	A 2D matrix of state-action pairs in each time step of an episode that shows the history of the agent's activities (Equation 1).
V Function	$\mathcal{V}$	A function that gets a state and returns the expected cumulative reward the agent can receive starting from that state (Equation 5 and Equation 9).

- $\mathcal{V}(s)$: State-value function (expected cumulative reward from state s).
- $\mathcal{R}(s,a)$: Immediate reward function.
- $\mathcal{P}(s'|s,a)$: Probability of transitioning from state s to s' when action a is taken.
- a_t: Action at time step t.
- s_t: State at time step t.
- $\mathcal{G}_\tau(t)$: Return value (cumulative discounted reward after time t in trajectory τ).
- τ: Trajectory (sequence of state-action pairs over an episode).
- $\rho(s)$: Probability distribution of the initial state.

Moreover, a step-by-step tutorial on how to design an RL model is also explained in this section.

A. Reinforcement Learning background

Before exploring the background of reinforcement learning, we will limit our discussion to episodic, model-free reinforcement learning, as the research reviewed in this paper falls into this category. However, to have an overview of different types of RL, we briefly describe them. Model-free RL focuses on learning based on trial and error fundamentals [8], while model-based RL tries to figure the environment model out and plan for performing actions with long-term values [9, 10]. About the learning performance of RL, we can say that the performance of an RL model can be measured in two main ways: episodic and continual (i.e., non-stationary). In episodic reinforcement learning, the number of episodes can be specified, and training for the prescribed duration enables the model to make optimal decisions[11]. This type of RL model is commonly used in situations where the termination of an episode can be specified. In other words, the training phase ends after a finite number of steps have been completed. On the other hand, the continual RL models do not have a specific termination state [12]. As a result, a discount factor is used to measure the learning performance, and once it exceeds the lowest acceptable value, the training phase is finished. Although there is no exact distinction between different RL categories, focusing on episodic model-free RL from now on, we will refer to this category simply as RL.

RL is a type of machine learning approach in which an agent learns how to make decisions and solve problems with the help of trial and error fundamentals. In other words, if the performed actions result in positive rewards, the agent is directed toward the best solution. On the other hand, in cases where negative rewards (i.e., punishments) are encountered, the agent realises that the strategy needs to be changed [13].

When the training phase is initiated, the agent starts exploring the problem environment, which is split into different states. The observation state set is $\mathcal{S}$. Each action the agent decides to perform among the action space ($\mathcal{A}$) leads to a state transition. The state transition probability is commonly

Algorithm 1 The procedure of training an RL model

```
Require: the learning parameters
Require: the environment parameters
Ensure: the trained model
 1: $M \leftarrow$ an instance of RL models
 2: Initiate $M$ based on the input parameters
 3: $A \leftarrow$ the agent of $M$
 4: while the learning performance of $A$ is not achieved do
 5:     while the $A$'s current episode is not terminated do
 6:         $s \leftarrow$ the current environment state
 7:         $a \leftarrow$ the $A$'s optimal action based on $s$
 8:         $r \leftarrow$ the obtained reward based on $(s, a)$
 9:         Perform state transition
10:         Update $A$ based on $(s, a, r)$
11:     Reset the environment
12: return $M$
```

Remark: The agent A and its parameters are encapsulated within the model M. Each update to A in Line 12 directly updates the parameters of M. Therefore, the returned M is the fully trained RL model that incorporates all updates performed during the training process.

shown as $\mathcal{P}(s \mid s_t, a_t)$, which is defined as the probability of transitioning to state s, when the agent takes action a_t at state s_t [14]. Different actions in different states result in different reward values. The reward function is commonly shown by $\mathcal{R}(s_t, a_t)$, where s_t is the current state and a_t is the selected action to be taken in that state. The trajectory from the initial state to the terminal state during the training phase is called an episode [15]. Once the terminal state is reached, a new episode starts after the environment state is reset to its initial. During these episodes, the agent learns how to make decisions and solve problems by trying to maximize the cumulative reward. The trajectory followed by the agent is denoted as τ, and it is defined as a sequence of state-action pairs, or, more precisely, a row matrix of column matrices. The matrix in the t^{th} column of τ indicates the state and the action in time step t. The first row of this element (s_t^τ) is the state, and the second row is the action (a_t^τ). The number of time steps in an episode with trajectory τ is also shown as $|\tau|$. The structure of a trajectory is shown in Equation 1

$$\tau = \begin{bmatrix} \begin{bmatrix} s_1^\tau \\ a_1^\tau \end{bmatrix} & \begin{bmatrix} s_2^\tau \\ a_2^\tau \end{bmatrix} & \dots & \begin{bmatrix} s_{|\tau|}^\tau \\ a_{|\tau|}^\tau \end{bmatrix} \end{bmatrix} \tag{1}$$

The probability of obtaining a trajectory is shown as $\mathcal{P}(\tau \mid \pi)$ and it is calculated based on Equation 2.

$$\mathcal{P}(\tau \mid \pi) = \rho(s_1^\tau) \prod_{t=1}^{|\tau|} \mathcal{P}(s_{t+1}^\tau \mid s_t^\tau, a_t^\tau) \pi(a_t^\tau \mid s_t^\tau) \tag{2}$$

In this equation, $\rho(s_1^\tau)$ is the probability of the first state of τ to be the initial state of the environment.

The general steps of the agent training phase in RL models are presented in Algorithm 1.

The learning parameters, such as the conditions of terminating the training phase and the environment parameters containing the information required for presenting it to the agent, are passed to this algorithm, and the trained model is its output. In the first step, the RL model is initiated and its parameters are set (Line 1 to Line 2). For simplicity, the agent of this model is stored in A. Then, a loop is considered for repeating the explorations until the agent learns the environment with a specified learning performance (Line 4 to 11). In this loop, another loop is considered for completing a single episode (Line 4 to Line 16). Then, the environment is reset to start again with the initial state (Line 11).

Remark: In Algorithm 1, the agent A and its parameters (such as policy weights or value functions) are encapsulated within the model M. Thus, each update to A in Line 12 effectively updates the parameters of M. As a result, the returned model M represents the fully trained RL model that incorporates all updates performed throughout the training episodes. This ensures that the output is the final version of the model, reflecting the cumulative learning process.

In the inner loop, dedicated to a single episode, the current state of the environment is generated (Line 6), and then the agent determines the optimal action to perform in this state (Line 7). The chosen action and the current state are then used for calculating the reward (Line 8). Next, the processes related to transitioning to the next state are done, and finally, the agent is updated based on the triple of state, action, and reward to improve itself.

In Line 7 of Algorithm 1, the agent decides an action to perform, and this decision-making is based on the probabilities, which are denoted as π, assigned to each action in that state. Hence, π indicates the probability of each action to be selected. If the agent follows a specific policy to select the actions, π is parameterized, and it is shown as π_θ, where θ is its parameters. $\pi_\theta(a \mid s)$ is interpreted as the probability that the agent following policy π with parameters θ, selects action a in state s. Three terms are commonly used in this step: return, state, and action value. Return value is the cumulative discounted reward the agent obtains after a specific time step. A parameter is used to calculate the return value, which is called the discount factor (γ), and indicates the importance of future rewards over the immediate reward (i.e., the output of $\mathcal{R}$). This parameter is between zero and one, and the higher values of γ show the higher importance of future rewards [16]. The return value of trajectory τ in time step t is shown as $\mathcal{G}_\tau(t)$ and calculated based on Equation 3.

$$\mathcal{G}_\tau(t) = \sum_{i=0}^{\infty} \gamma^i R(s_{t+i}^\tau, a_{t+i}^\tau) \tag{3}$$

In the case that there is a finite number of time steps, say T, in the related episode, the return value is defined as Equation 4.

$$\mathcal{G}_\tau(t) = \sum_{i=0}^{|\tau|-t} \gamma^i R(s_{t+i}^\tau, a_{t+i}^\tau) \tag{4}$$

State value is the expected value (i.e., cumulative reward) the agent obtains starting from a specific state. The state value function, which is also called V function or simply V function, is shown as $\mathcal{V}_\pi(s)$ and it is defined based on Equation 5 [17], where s is that specific state we want to calculate its value and γ is the discount factor. The discount factor is a parameter that

$$\mathcal{V}_\pi(s) = \sum_{a' \in \mathcal{A}} \pi(a' \mid s) \left(\mathcal{R}(s,a') + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s' \mid s,a') \mathcal{V}_\pi(s') \right) \tag{5}$$

On the other hand, Action value, which is also known as Q value, determines the performance of an action in a state. In other words, it is the value (i.e., cumulative reward) of taking a specific action within a particular state, and its related function is shown as $\mathcal{Q}_\pi(s,a)$, where s and a are that state and action, respectively. The definition of the Q function, which is also known as the action-value function, is presented in Equation 6.

$$\mathcal{Q}_\pi(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s' \mid s,a) \mathcal{V}_\pi(s') \tag{6}$$

Now, let us analyze the definition of the V and Q functions. As the outermost parentheses of the right side of Equation 5 is equal to Equation 6, the Q function is explained first. When the agent takes action a in state s, an immediate reward ($\mathcal{R}(s,a)$) is received, and then, we expect some other rewards in future. The future rewards are the probability-weighted sum of the rewards in future states. The likelihood of each state, say s', to be the next state after a is taken in s is $\mathcal{P}(s' \mid s,a)$, and its value is $\mathcal{V}_\pi(s')$. Hence, adding the product of these probabilities and their related state value gives us the expected future cumulative reward. Now, to consider the importance of future rewards over the immediate reward, we multiply the sum of future rewards by the discount factor (γ). Finally, the action value is calculated by adding the immediate reward and the discounted future rewards, which is represented on the right side of Equation 6. Furthermore, the expected value of state s can be calculated as the probability-weighted sum of all possible actions in s. Each action, say a, is selected in s by the probability of $\pi(a \mid s)$, and the value of that action is its action value. Therefore, adding the product of these probabilities and their related value for all possible actions gives us the state value, which is defined in Equation 5.

The optimal action and state values, which are determined when the optimal policy or any trajectory that leads to the maximum value is followed, are shown as $\mathcal{Q}_*(s,a)$ and $\mathcal{V}_*(s)$, respectively. To calculate the optimal action value, we must maximise the action value, which is equivalent to maximising the right side of Equation 6. The only expression on this side that can be maximised is the value of $\mathcal{V}_\pi(s')$. Hence, by replacing it with its maximum ($\mathcal{V}_*(s')$), we reach Equation 7.

$$\mathcal{Q}_*(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s' \mid s,a) \mathcal{V}_*(s') \tag{7}$$

On the other hand, for maximising the state value and getting $\mathcal{V}_*$, we have to maximise the right side of Equation 5. Similar to the process of calculating the optimal action value, the expression inside the outermost parenthesis of this side can be maximized by replacing $\mathcal{V}_\pi(s')$ by $\mathcal{V}_*(s')$. Moreover, for maximizing the whole expression in the right side of this equation, since the value of $\pi(a',s)$ is policy-specified, we can first find the optimal action, say a'', that maximizes the outermost parentheses, and then consider a policy (i.e., distribution) that selects it by the probability of one ($\pi(a'',s) = 1$). Hence, we can define the optimal state value function as Equation 8.

$$\mathcal{V}_*(s) = \max_{a' \in \mathcal{A}} \left(\mathcal{R}(s,a') + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s' \mid s,a') \mathcal{V}_*(s') \right) \tag{8}$$

It is worth noting that in Equation 5, Equation 6, Equation 7, and Equation 8 the values are calculated based on future state values. However, these equations can be written based on the action values too. They are shown in Equation 9, Equation 10, Equation 11, and Equation 12, respectively.

$$\mathcal{V}_\pi(s) = \sum_{a' \in \mathcal{A}} \pi(a' \mid s) \mathcal{Q}_\pi(s,a') \tag{9}$$

$$\mathcal{Q}_\pi(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s' \mid s,a) \sum_{a' \in \mathcal{A}} \pi(a' \mid s') \mathcal{Q}_\pi(s',a') \tag{10}$$

$$\mathcal{Q}_*(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s' \mid s,a) \max_{a' \in \mathcal{A}} \mathcal{Q}_*(s',a') \tag{11}$$

$$\mathcal{V}^*_\pi(s) = \max_{a' \in \mathcal{A}} \mathcal{Q}_*(s,a') \tag{12}$$

The agent will improve its learning performance based on the received rewards (Line 12 in Algorithm 1). An ideal agent knows the optimal action in each state. The optimal action in each state is the one that maximizes the action value (Equation 13).

$$\text{Optimal Action in } s = \arg\max_{a' \in A} \mathcal{Q}_\pi(s,a') \tag{13}$$

The optimal policy is also the one that results in the highest state value in each state. Since the policy is defined by its parameters, the optimal θ is sought (Equation 14).

$$\text{Optimal Policy Parameters in } s = \arg\max_{\theta'} \mathcal{V}_{\pi_{\theta'}}(s) \tag{14}$$

However, in most problems aimed to be solved by RL, finding the optimal action or policy is not straightforward, due to the high computational complexity of the related equations. Hence, different approaches exist for updating the agent in this step. The difference between these approaches is in their focus on optimal actions or optimal policies. In this regard, they can be classified into value- and policy-based approaches. The value-based approaches focus on optimizing the actions, while the policy-based methods optimize the policy. These approaches can also be categorized into on-policy and off-policy. In on-policy methods, the updated feature, which may be values of policies, is improved following the same π/policy. However, in off-policy methods, a different π/policy is used during updates to improve performance. The commonly used updating methods using these approaches are explained as follows:

- **Q-Learning**: In the Q-Learning method, which has a value-based approach, the agent starts with the initial action values (i.e., Q values), and then updates them in each exploration step. At the end of the learning process, the agent can predict the action values, and the optimal action is found based on them. Two of the most common Q-learning techniques are as follows:

– Vanilla Q-Learning: In the Vanilla Q-Learning technique, the agent uses a table, called Q table, the rows and columns of which denote the states and the actions, respectively, and the action values are stored in its cells. The agent extracts the related row from the Q table in each state. The column with the highest action value in this row indicates the action that maximizes the action value. When the agent is in state s_t, performs action a, and receives reward r, and the environment state is transitioned into s_{t+1}, the new action value is the probability-weighted sum of the old action value and the newly learned one. As a result, one can write the update rule of the Vanilla Q-Learning technique as Equation 15. The probability weight factor in this updating process is called the learning rate and is shown as α. The learning rate indicates the proportion of newly learned knowledge that must be used to update the general learned knowledge, which is typically between zero and one.

$$Q(s_t, a) \leftarrow (1-\alpha)Q(s_t, a) + \alpha(r + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}, a')) \tag{15}$$

– Deep Q-Learning: The agent in Deep Q-Learning techniques uses deep neural networks, which are also called Deep Q-Network (DQN), to update the values and cover the scalability limitations of Vanilla Q-Learning, such as high memory usage for storing the Q table and the huge number of state-action pairs in large-scale problems. A DQN consists of an input layer, an output layer, and one or more hidden layers, and the layers also consist of neurons. Each neuron defines a parameterized function that gets the neuron input and generates the neuron output. This function is known as the activation function, and its parameters are also referred to as weights. The neurons in the input layer receive the network inputs and pass the returned value to the first hidden layer. The neurons in the last hidden layer receive the input layer's returned values, in the case of a single hidden layer, or the previously hidden layer's returned value in other cases, and pass the returned value to the output layer. The output layer neurons receive them, and produce the final output. During the training phase of a DQN, the weights are updated so that the network's final output is close to the expected value (i.e., the truth value). For this reason, a loss function that determines the difference between the neural network output and the truth value is considered and minimized during the training phase. In the Deep Q-Learning technique, the DQN gets the states as its input and returns the actions and their action values. There are $|\mathcal{A}|$ neurons in the output layer of this network, and the output of these neurons is the action value of the related action. The loss function of Deep Q-Learning is shown in Equation 16, where $\mathcal{L}(s_t, a, s_{t+1})$ is the output of the loss function when the agent is in state s_t and takes action a, and the state is transitioned into s_{t+1}.

$$\mathcal{L}(s_t, a, s_{t+1}) = \left| r + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}, a') - Q(s_t, a) \right|^2 \tag{16}$$

It is worth noting that Q-Learning is an off-policy algorithm because the actions selected do not always follow the same distribution, as the values change through the improvement steps.

- **Policy Gradient**: Unlike the Q-Learning method, the Policy Gradient method focuses on learning the optimal policy instead of the optimal actions. The main idea behind this type of method is to run an episode first, collect the obtained trajectory, and update the policy parameters (θ) based on it and some derivative rules in calculus. Repeating this process helps the agent learn the optimal policy. An objective function ($\mathcal{J}(\theta)$) is defined, its derivative in terms of θ ($\nabla_\theta \mathcal{J}(\theta)$) is calculated, and then θ is updated based on it. The commonly used techniques for the Policy Gradient method are as follows.

– Vanilla Policy Gradient: The Vanilla Policy Gradient technique defines the objective function as the expected return. It starts with running $|T|$ episodes, the trajectories of which are stored in T. Then the gradient of the expected return is calculated based on Equation 17.

$$\nabla_\theta \mathcal{J}(\theta) = \sum_{\tau \in T} \sum_{t=1}^{|\tau|} \nabla_\theta \log \pi_\theta(a_t^\tau \mid s_t^\tau) \mathcal{G}_\tau(1) \tag{17}$$

After calculating the gradient, the policy parameters are updated according to Equation 18.

$$\theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{J}(\theta) \tag{18}$$

Repeating this process leads to the optimal policy. It is worth noting that the policy is randomly initiated, and for subsequent repeats, the previously updated policy is used to generate the trajectories. We can see that the Vanilla Policy Gradient is on-policy, as the policy used for generating the trajectories is the same as the one being updated.

– REINFORCE: The REINFORCE technique runs an episode completely, and then, updates its policy based on Equation 19 in each time step (t) of that trajectory (τ).

$$\theta \leftarrow \theta + \alpha \mathcal{G}_\tau(t) \nabla_\theta \log \pi(a_t^\tau \mid s_t^\tau) \tag{19}$$

This process is repeated and the trajectory of each episode follows the updated policy. During these repeats, the agent learns the optimal policy. The REINFORCE technique is on-policy since a complete trajectory is generated, and then its policy is updated.

- Actor-Critic: This method uses two main updaters for improving its learning performance: actor and critic. The actor updater is responsible for deciding which action to take by learning the policy. In contrast, the critic updater demonstrates the efficiency of the performed action and how it can be improved by focusing on the gained values [18]. One can look at the Actor-Critic technique as a combination of a policy-based approach (i.e., actor) and a value-based approach (critic). In this technique, an entire episode is executed, followed by another pseudo-episode that starts from the initial state of the main episode and performs an action based on the Policy/Distribution of the main episode. This process is repeated until the pseudo-episode is finished. At each time step of this pseudo-episode, the actor and critic update the policy parameters and values, respectively, based on Equation 20 and Equation 21.

$$\theta \leftarrow \theta+\alpha_\theta(\mathcal{R}(s,a)+\gamma\mathcal{Q}_\omega(s',a')-\mathcal{Q}_\omega(s,a))\nabla_\theta \log \pi_\theta(a|s) \tag{20}$$

$$\omega \leftarrow \omega+\alpha_\omega(\mathcal{R}(s,a)+\gamma\mathcal{Q}_w(s',a')-\mathcal{Q}_w(s,a))\nabla_\theta \mathcal{Q}_\omega(s,a) \tag{21}$$

There are different action selection techniques defined for RL agents, the most common of which are as follows [19]:

- ϵ-Greedy action selection: Following this technique, the agent acts with the highest action value most of the time, and at other times a random action is selected. In other words, the action with the highest action value is more likely to be selected, but the probability of other actions is the same. ϵ in this technique represents a small, positive number less than one, indicating the probability that the currently best action will not be selected.
- Softmax action selection: Unlike the ϵ-greedy technique, the Softmax technique does not treat actions other than those with the highest action value the same. All the actions are ranked based on their current action value, and the probability of selecting them is formed regarding this ranking. Boltzmann [20] is one of the most commonly used distributions for Softmax. This distribution is based on Equation 22, where ψ is a parameter known as the temperature parameter, which indicates how actions with different probabilities and action values are treated. Higher values of ψ make the probability of different actions to be selected nearly the same. On the other hand, lower values of this parameter cause a bigger difference between the selection probability of the actions with different action values. As much as ψ is near to zero, Softmax becomes more similar to ϵ-Greedy.

$$\pi(a \mid s) = \frac{e^{Q(s,a)/\psi}}{\sum\limits_{a\in\mathcal{A}} e^{Q(s,a)/\psi}} \tag{22}$$

B. RL design

Now, we give a sample scenario to show the details of each step in Algorithm 1. This scenario is described in Scenario 1 and visualized in Figure 2.

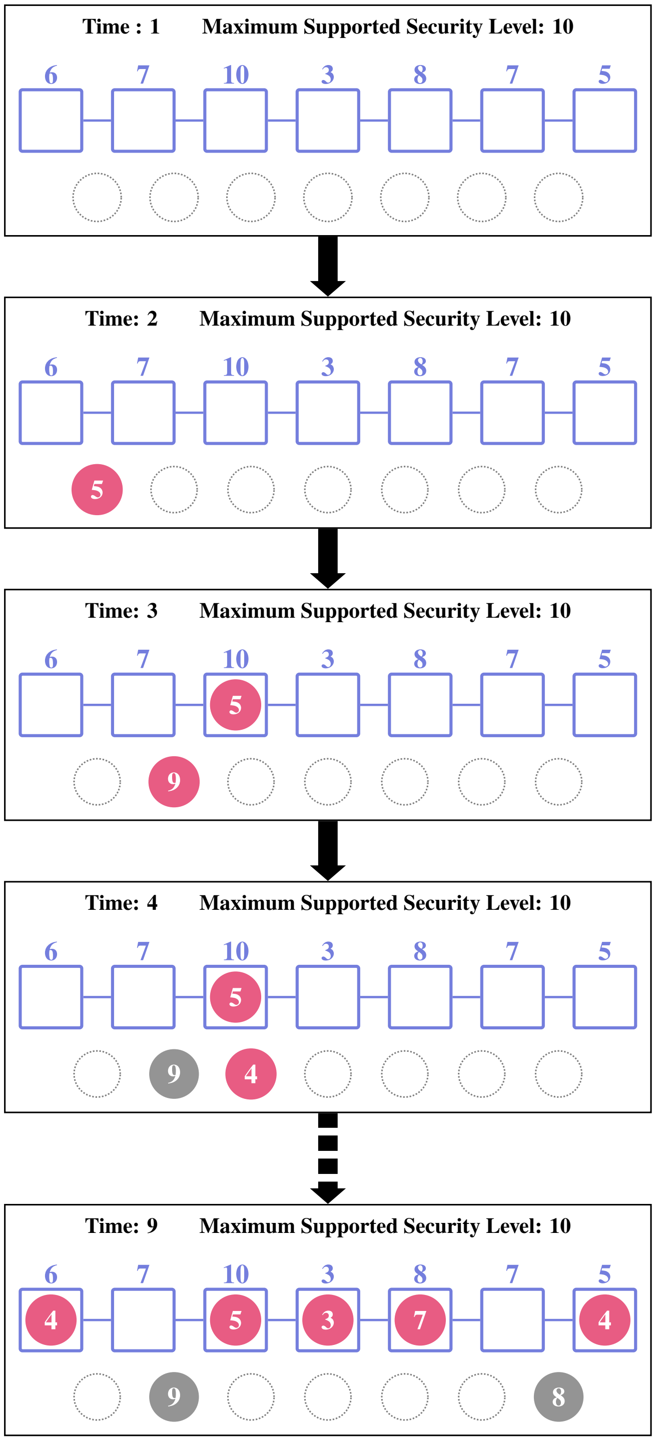
Scenario 1. *A resource provider has a network of seven physical nodes, and users can request the mapping of their requests as virtual nodes on these resources. Ten different security levels are defined for the physical resources ($xsec$ = 10), which are indicated by integer values, where 1 is the lowest and 10 is the highest security level, respectively. The physical resources are not always the same, and they may differ in the maximum security level they can support. Moreover, since the resource provider updates the physical resources or some of them are downgraded through time, their supported security level is not fixed. Each of the physical resources can support different security levels. For example, if the security level of a resource is μ, it can support all the requests that require security levels up to μ. A request for mapping a virtual node on a physical resource is considered accepted if and only if that resource can support its requested security level and there are no previously mapped requests on that physical node. Furthermore, the number of arrival requests equals the number of physical resources (i.e., seven).*

Algorithm 2 The procedure of initiating Δ_{Naive}

```
Require: the learning parameters ($lr$)
Require: the environment parameters ($env$)
Ensure: the learning model instance ($M$)
 1: $M \leftarrow$ an instance of RL models
 2: $M.agent \leftarrow$ an instance of RL agents
 3: $M.agent.e \leftarrow lr.episodes$
 4: $M.agent.a \leftarrow lr.alpha$
 5: $M.agent.g \leftarrow lr.gamma$
 6: $M.agent.p \leftarrow lr.psi$
 7: $M.agent.n \leftarrow env.N$     ▷ the no. of physical resources
 8: $M.agent.xsec \leftarrow env.xsec$ ▷ the max. supported levels
 9: $M.agent.q \leftarrow$ an empty dictionary
10: $M \leftarrow$ Algorithm 3 with $M$ as its input
11: return $M$
```

Considering Scenario 1, we are going to define an RL model, called Δ_{Naive}, that can optimally map the requests on the resources. This model aims to map as many requests as possible on the physical resources. The first step is to initiate the Δ_{Naive}, which is shown in Algorithm 2. The RL agent stores the learning parameters to update itself in the training phase. As a result, the number of episodes ($episodes$) and the values of α ($alpha$), γ ($gamma$), and ψ (psi), which are provided as the learning parameters, are stored on the agent (Line 3 to Line 6). Then, the required information of the environment is extracted (Line 7 and Line 8). This information is the number of physical resources (n) and the maximum supported security levels by the resources ($xsec$). Then, the initial Q table, a dictionary that gets a state and an action, and returns the action value, is generated and stored in q of the agent (Line 9). Finally, the generated model is reset (Algorithm 3) for the first time and returned.

The procedure of resetting Δ_{Naive} is presented in Algorithm 3. All the required operations, which have to be performed before an episode is started, are contained in the process of resetting Δ_{Naive}. Moreover, the agent has to be trained so that the optimal solution in different environments, but with the same configurations as Scenario 1, can be found. As a result, some parameters are randomly set, and in each episode, they are changed. Using these random parameters, the agent is generally trained without dependency on the variable numbers. Keeping this generality in mind, the security level of both physical and virtual nodes is randomly generated. A loop with n rounds is considered in this procedure, in which



Time : 1 Maximum Supported Security Level: 10
6 7 10 3 8 7 5
Time: 2 Maximum Supported Security Level: 10
6 7 10 3 8 7 5
5
Time: 3 Maximum Supported Security Level: 10
6 7 10 3 8 7 5
5
9
Time: 4 Maximum Supported Security Level: 10
6 7 10 3 8 7 5
5
9 4
Time: 9 Maximum Supported Security Level: 10
6 7 10 3 8 7 5
4 5 3 7 4
9 8

Fig. 2. Visualization of Scenario 1 with sample supported and requested security levels, where the squares and circles illustrate the resources' physical nodes and the requests' virtual nodes, respectively. The numbers inside the circles and on top of squares indicate the requested and supported security levels, respectively, and the gray circles are the rejected requests (i.e., the requests that are not mapped on the resources due to the constraints).

two random numbers between one and $xsec$ are generated to indicate the security levels of the physical and virtual nodes (Line 4 and Line 7). It must be noted that the agent does not access the requested information, except that of the currently arrived one at each time. If the agent is aware of all the

Algorithm 3 The procedure of resetting Δ_{Naive}

```
Require: the learning model instance ($M$)
Ensure: the reset model ($M$)
 1: $M.agent.sec \leftarrow$ an empty list
 2: $M.req \leftarrow$ an empty list
 3: for $1 \leqslant i \leqslant M.agent.n$ do
 4:     $sec \leftarrow$ random$(1, M.xsec)$
 5:     $M.agent.sec \leftarrow M.agent.sec + \{sec\}$
 6:     $req \leftarrow$ random$(1, M.agent.xsec)$
 7:     $M.req \leftarrow M.req + \{req\}$
 8: $M.agent.r \leftarrow 1$                    ▷ the current request
 9: $M.agent.map \leftarrow$ a list of $M.agent.n$ $NULL$s
10: return $M$
```

Algorithm 4 The procedure of generating the environment states in Δ_{Naive}

```
Require: the learning model instance ($M$)
Ensure: the current state ($s$)
 1: $A \leftarrow M.agent$
 2: $s \leftarrow$ a list of $A.n$ pairs of zeros
 3: for $1 \leqslant i \leqslant A.n$ do
 4:     $s[i][1] \leftarrow \{A.sec[i]\}$
 5:     if $A.sec[i] < M.req[A.r]$ then
 6:         $s[i][2] \leftarrow \{0\}$
 7:     else if $A.map[i] \, != NULL$ then
 8:         $s[i][2] \leftarrow \{0\}$
 9:     else
10:         $s[i][2] \leftarrow \{1\}$
11: return $s$
```

arrived requests, it can sort and map them more effectively. However, the agent is unaware of future requests in the real world. As a result, the agent must be trained without complete information on the requests, which is why this information is stored in the model itself. The agent also stores a pointer that must be initiated at each episode's beginning. r points to the current request that must be processed and mapped. Since an episode is started by processing the first request, this pointer has to be initiated with one (Line 8). A list (map) for storing the mapping of virtual nodes to physical resources is also initialised with $NULL$ values by the agent (Line 9). This list's i^{th} element is the virtual node mapped on the i^{th} physical resource. If this element is $NULL$, no virtual nodes are successfully mapped on that physical node.

To present the environment to the agent, we use two features for each resource: the security level and the capability of hosting the current virtual node. To be more precise, we define the environment state as Equation 23. The algorithm for generating the environment states is presented in Algorithm 4. To form the current state, we need a list with n pairs of zeros (Line 2), which will be updated during this algorithm. At the final step of this algorithm, the i^{th} element of this list is a pair, the first part of which ($s[i][1]$) indicates the security level of the i^{th} physical node, and the second part ($s[i][2]$) specifies whether or not the current request can be mapped on the i^{th} resource. We loop over all the resources and put their security level at the first part of the related pair (Line 4).

Then, we must verify the capability of each resource to host the current virtual node. So, we check three conditions. The first check is on the security level constraint. If the current resource does not support the requested security level, a zero is put in the second part of the current pair (Line 5). If the security level constraint is satisfied, the next step is to check whether the current resource has a previously mapped virtual node (Line 7). If this constraint is also satisfied, the second part of the current pair becomes one (Line 9).

Model 1: Δ_{Naive}

$$\mathcal{S} = \begin{bmatrix} sec_1 & sec_2 & \dots & sec_N \\ vd_1 & vd_2 & \dots & vd_N \end{bmatrix} \tag{23}$$

$$\mathcal{R} = \begin{cases} 0, & \nexists i \in \mathbb{N}_N : vd_i = 1 \\ -1, & vd_a = 0 \textit{ and } \exists i \in \mathbb{N}_N : vd_i = 1 \\ \dfrac{\sum_{i=1}^{N} vd_i \times h_{i,a}}{\sum_{i=1}^{N} vd_i}, & vd_a = 1 \end{cases} \tag{24}$$

N	:	*The total number of physical resources*
$\mathbb{N}_N$	:	*The set of natural numbers lower than $N+1$*
a	:	*The selected action*
sec_i	:	*The supported security level of the i^{th} physical node*
vd_i	:	*A binary value, which is one if the current virtual node can be mapped on the i^{th} physical node*
$h_{i,a}$	:	*A binary value, which is one if the i^{th} resource has a security level higher than or equal to the selected action (a)*

The actions in Δ_{Naive} are defined as the physical node the agent suggests for mapping the current virtual node. For example, if the agent's first action is two, the first virtual node (i.e., the first request) is mapped on the second physical resource. Hence, the action space size of Δ_{Naive} is n, which is equal to seven for Scenario 1. The agent in Δ_{Naive} selects the optimal action based on Boltzmann, which is described in Algorithm 5. Before explaining this algorithm, let's use an example to select an event with a specific distribution. Suppose that there are three events and their probability of occurrence is $\{0.1, 0.4, 0.5\}$. To generate a random number based on these probabilities, we first need to create a list of their cumulative sums. This list contains four elements, starting with zero. The next elements are the partial summations. In our example, the cumulative sum is $\{0, 0.1, 0.1 + 0.4, 0.1 + 0.4 + 0.5\}$, which is equivalent to $\{0, 0.1, 0.5, 1.0\}$. Now, if we generate a float number between zero and one, and its value is higher than the i^{th} element and lower than or equal to the $(i+1)^{th}$ element of this list, the chosen event is the i^{th} one. For example, if the random number is 0.5, we have $0.1 < 0.5 \leqslant 0.5$, and hence, the second event is selected. This technique is used in Algorithm 5. First, variables and lists for storing the partial summations (sum) and the cumulative sum (cu), respectively, are initialised (Line 1 and Line 2). Then, cu is filled through a loop (Line 3 to Line 7). It must be noted that since the action values are initialized with zero, if the entry of the current state and a specific action ((s, i)) is not found in the dictionary that stores the action values (q), it is considered to be zero (Line 4). Then a random float number between zero and one is generated (Line 8). After that, another loop is used to find the location of this random number among the cumulative sum elements (Line 9 to Line 11).

Algorithm 5 The procedure of action selection in Δ_{Naive} based on the Boltzmann distribution

```
Require: the learning model's agent instance ($A$)
Require: the current state ($s$)
Ensure: the optimal action ($a$)
 1: $sum \leftarrow 0$
 2: $cu \leftarrow \{0\}$
 3: for $1 \leqslant i \leqslant A.n$ do
 4:     if $(s, i)$ is not defined on $A.q$ then
 5:         $A.q(s, i) \leftarrow 0$
 6:     $sum \leftarrow sum + \mathbf{exp}(A.q(s, i)/A.p)$
 7:     $cu \leftarrow cu + \{sum\}$
 8: $ra \leftarrow$ float_random$(0, 1)$
 9: for $1 \leqslant a \leqslant A.n$ do
10:     if $cu[a] < ra \leqslant cu[a + 1]$ then
11:         break
12: return $a$
```

Based on the defined action space and observation states for Δ_{Naive}, we have defined the reward function as Equation 24. If there is no valid action to perform, the agent is neither punished nor rewarded, but if the mapping is invalid, the agent is punished. Otherwise, the reward is the ratio of the number of physical resources with a security level higher than or equal to that of the chosen one by the agent to the total number of resources that are valid for mapping the current request. Defining the reward function in this way can perfectly lead the agent toward the solution. In the cases that there are no possible mapping solutions, the agent receives zero as the reward to avoid misleading rewards. When a wrong action (i.e., an action causing an unsuccessful mapping) is performed, the agent is punished and hence, trained to avoid them. On the other hand, if a request is mapped on a resource with the minimum possible supported security levels, we have saved our resources. For example, suppose that there are three resources with levels as $\{1, 4, 7\}$, and a user requests for a security level of three. If this request is mapped on the third resource (i.e., the one with seven supported levels), and another user asks for five levels, the second request cannot be mapped. However, if we map the first request on the second resource, we can map the second request on the last resource. Therefore, a higher reward is received when the agent selects the resource with the lowest possible levels. The other point about this function is that when a positive reward is given to the agent, the number of resources with higher or equal security levels must be divided by the total number of capable resources. This is because the agent's performance depends entirely on the environment's situation. We give an example to demonstrate it better. Assume two situations. In the first one,

Algorithm 6 The procedure of calculating the reward function in Δ_{Naive}

```
Require: the learning model's agent instance ($A$)
Require: the current state ($s$)
Require: the selected action ($a$)
Ensure: the reward ($r$)
 1: $valid \leftarrow 0$
 2: $high \leftarrow 0$
 3: for $1 \leqslant i \leqslant A.n$ do
 4:     if $s[i][2] = 1$ then
 5:         $valid \leftarrow valid + 1$
 6:     if $s[a][1] \leqslant s[i][1]$ then
 7:         $high \leftarrow high + 1$
 8: if $valid = 0$ then
 9:     $r \leftarrow 0$
10: else if $s[a][2] = 0$ then
11:     $r \leftarrow -1$
12: else
13:     $r \leftarrow high\ /\ valid$
14: return $r$
```

Algorithm 7 The procedure of checking if the current episode is terminated in Δ_{Naive}

```
Require: the learning model's agent instance ($A$)
Ensure: the episode termination flag ($done$)
 1: if $A.r > A.n$ then
 2:     $done \leftarrow 1$
 3: else
 4:     $done \leftarrow 0$
 5: return $done$
```

there are only two valid resources, and the agent selects the one with the lowest level. In the second situation, there are six valid resources, and the agent selects the penultimate resource from the sorted list of resources based on their levels. In both of these situations, the number of resources with higher or equal level, considering the selected source itself, is two. But, the agents perform more efficiently in the first situation. As a result, the reward is defined as a fraction.

The procedure of calculating the reward function in Δ_{Naive} is shown in Algorithm 6. First, two variables are considered for storing the total number of valid resources ($valid$) and those that have a higher or equal level compared to the selected resource ($high$), and they are initialized with zero (Line 1 and Line 2). Then a loop through all the resources is considered, in which the values of $valid$ and $high$ are updated (Line 3 to Line 7). At last, three conditions are checked that are equivalent to the cases defined in Equation 24, and the obtained reward is stored in r (Line 8 to Line 13). Finally, r as the reward is returned.

The agent has to know the terminal state of the environment. In our case, the terminal state is when all the requests are processed. Hence, we have Algorithm 7 for checking if an episode is terminated.

Once the agent selects and updates an action, the current virtual node is mapped on the chosen physical node, and the related variable for storing the mappings is also updated.

Algorithm 8 The procedure of state transition in Δ_{Naive}

```
Require: the learning model instance ($M$)
Require: the selected action ($a$)
Ensure: the updated agent ($A$)
 1: $A \leftarrow M.agent$
 2: if $A.sec[a] \geqslant M.req[A.r]$ then
 3:     if $A.map[a] \neq NULL$ then
 4:         $A.map[a] \leftarrow A.r$
 5: $A.r \leftarrow A.r + 1$
 6: return $A$
```

Algorithm 9 The procedure of updating the agent in Δ_{Naive} based on Vanilla Q-learning

```
Require: the learning model's agent instance ($A$)
Require: the current and the next states ($s$ and $s'$)
Require: the selected action ($a$)
Require: the obtained reward ($r$)
Ensure: the updated agent ($A$)
 1: $x \leftarrow 0$
 2: for $1 \leqslant i \leqslant A.n$ do
 3:     if $A.q[s'][i] > x$ then
 4:         $x \leftarrow A.q[s'][i]$
 5: $x \leftarrow (r + A.g * x)$
 6: $A.q[s][a] \leftarrow (1 - A.a) * A.q[s][a] + A.a * x$
 7: return $A$
```

Furthermore, the pointer pointing to the current request is incremented to indicate the start of processing the next request. The details of this state transition process are shown in Algorithm 8. For state transitions, three main operations must be performed. The first operation involves updating the agent's mapping list if the mapping is successfully completed. To check the validity of the mapping process, the security level and the condition of hosting another virtual node are checked for the selected resource (Line 2 and Line 3). If all constraints are satisfied, the related element in the mapping list is set to the current request index (Line 4). The second operation is updating the current request pointer, which is done by incrementing it (Line 5).

Now that the states, actions, and rewards are specified, we can present the updating procedure of the agent in Δ_{Naive} (Algorithm 9), which utilises the steps of the Vanilla Q-learning method. The first step in this algorithm is to find the maximum value that can be achieved in the next state (s'). This value is stored in x. A loop over all possible actions is formed, and then the maximum value of the Q table is found (Line 2 to Line 4). Then the agent's Q table (q) is updated based on Equation 6 (Line 6), and the updated agent is returned.

In this step, all the procedures required for training Δ_{Naive} are defined, and we can customize Algorithm 1 for our model. The details are mentioned in Algorithm 10.

C. Related Works and Recent Advances (2019–2024)

In recent years, there has been a significant increase in research on applying reinforcement learning (RL), machine learning (ML), and artificial intelligence (AI) to various

Algorithm 10 The procedure of training Δ_{Naive}

```
Require: the learning parameters ($lr$)
Require: the environment parameters ($env$)
Ensure: the trained model ($M$)
 1: $M \leftarrow$ Algorithm 2 with $lr$ and $env$ as its inputs
 2: $episode \leftarrow 1$
 3: for $1 \leqslant episodes \leqslant M.agent.e$ do
 4:     while TRUE do
 5:         $A \leftarrow M.agent$
 6:         $s \leftarrow$ Algorithm 4 with $M$ as its input
 7:         $a \leftarrow$ Algorithm 5 with $A$ and $s$ as its inputs
 8:         $r \leftarrow$ Algorithm 6 with $A$, $s$, and $a$ as its inputs
 9:         $A \leftarrow$ Algorithm 8 with $M$ and $a$ as its inputs
10:         $M.agent \leftarrow A$
11:         $s' \leftarrow$ Algorithm 4 with $M$ as its input
12:         $A \leftarrow$ Algorithm 9 with $A$, $s$, $s'$, $a$, and $r$
13:         $M.agent \leftarrow A$
14:         $done \leftarrow$ Algorithm 7 with $A$ as its input
15:         if $done = 1$ then
16:             break
17:     $M \leftarrow$ Algorithm 3 with $M$ as its input
18: return $M$
```

network security challenges. Numerous high-impact studies published in journals such as JISAS, IEEE Transactions on Dependable and Secure Computing (TDSC), IEEE Transactions on Information Forensics and Security (TIFS), Elsevier Computer Networks, and IEEE Communications Surveys have proposed innovative RL-based approaches for problems including intrusion detection, DDoS mitigation, adversarial attack resistance, anomaly detection, and adaptive access control.

Table II provides an overview of representative recent works, summarizing their core techniques, targeted security challenges, and main outcomes. These advances demonstrate the growing maturity and diversity of RL-driven security research, further motivating a comprehensive and systematic survey of the field.

Recent works such as [21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37] have greatly advanced the application of RL in various network security scenarios, leveraging deep architectures, graph models, federated and adversarial learning, and multi-agent strategies.

D. Complementary Perspectives: Additional Recent Advances (2019–2024)

In addition to the mainstream advances in RL-driven network security, an expanding body of research has emerged in recent years exploring alternative and complementary perspectives. These studies move beyond classical network scenarios and address security challenges in a wide range of emerging domains, architectures, and threat landscapes. Notable areas of recent focus include:

- **Industrial IoT and Cyber-Physical Systems:** Reinforcement learning is playing an increasingly pivotal role in safeguarding industrial control systems (ICS) and industrial Internet of Things (IIoT) infrastructures. In these mission-critical environments, RL-based approaches enable the automation of adaptive access control, the early detection of sophisticated threats, and the mitigation of real-time anomalies. Advanced RL frameworks are used to dynamically adjust security policies in response to evolving attack patterns, ensuring operational continuity, minimizing false alarms, and protecting both physical assets and sensitive data from cyber-physical threats. The ability to learn from interactions within complex, high-dimensional industrial settings allows RL-driven systems to outperform traditional rule-based methods, particularly in scenarios such as smart factories, critical utilities, and automated production lines.
- **Blockchain and Distributed Ledger Integration:** Recent research has focused on the synergistic combination of RL and blockchain to enhance security in decentralized and collaborative environments. RL agents can be utilized to optimize consensus mechanisms, manage resource allocation, and detect malicious activities in distributed ledger systems. Integration of RL with blockchain strengthens secure data sharing, access control, and trust management across untrusted peers by autonomously learning optimal strategies for authentication, authorization, and anomaly response. These advancements are especially valuable in IoT networks, supply chain applications, and financial platforms where trust, traceability, and resilience are paramount.
- **Automotive and Connected Vehicle Security:** The rapid proliferation of connected vehicles and smart transportation systems has introduced unique security and safety challenges due to the high mobility, heterogeneous communication, and stringent latency requirements. RL-based security frameworks are now being designed to perform adaptive intrusion detection, secure routing, and cooperative defense against evolving threats in vehicular ad hoc networks (VANETs), vehicle-to-everything (V2X) communications, and autonomous driving platforms. Through continuous learning and adaptation, these RL-enabled systems are capable of identifying novel attack vectors, mitigating cyber-physical risks, and maintaining reliable service quality in real-time.
- **6G and Next-Generation Wireless Networks:** As wireless networks advance toward 6G, characterized by massive device connectivity, ultra-reliable low-latency communications, and pervasive AI integration, RL has emerged as a core enabler for context-aware and adaptive security management. RL algorithms support autonomous threat detection, dynamic resource provisioning, and policy optimization in ultra-dense, multi-access edge computing (MEC), and intelligent reflecting surface (IRS)-assisted networks. The flexibility of RL enables these next-generation systems to proactively adapt to changing network topologies, user behaviors, and complex attack surfaces with minimal human intervention.
- **Adaptive and Online Defense:** There is a growing emphasis on the development of security solutions that leverage online reinforcement learning and real-time anomaly detection. These systems can continuously monitor net-

TABLE II
REPRESENTATIVE RECENT WORKS ON RL/ML/AI FOR NETWORK SECURITY (2019–2024)

Reference	Venue/Year	Technique/Model	Target Security Problem	Key Contribution
Zhang et al. [21]	IEEE TDSC, 2022	Deep RL (DRL), GNN	Distributed intrusion detection in IoT	Robust, scalable DRL-based cooperative detection
Li et al. [22]	Computer Networks, 2022	Multi-agent DRL	Mitigation of DDoS in SDN	Distributed multi-agent DRL for attack mitigation
Kim et al. [23]	JISAS, 2021	Adversarial RL, GANs	Adversarial attack defense in 5G	RL-based GAN to defend against adversarial threats
Wang et al. [24]	IEEE TIFS, 2023	Federated RL	Privacy-preserving anomaly detection	Federated RL for scalable and privacy-aware detection
Patel et al. [25]	Elsevier CN, 2020	DRL for SDN	Adaptive access control	RL-based dynamic policy optimization
Chen et al. [26]	IEEE TIFS, 2024	Graph RL, GCN	Attack path prediction in cloud networks	GCN-enhanced RL for attack graph analytics
Singh et al. [27]	JISAS, 2023	Decentralized RL	IoT device authentication	Scalable RL-based authentication in IoT
Xu et al. [28]	IEEE Comms Surveys, 2023	Deep RL Survey	Security in SDN	Comprehensive analysis of deep RL-based security in SDN
Huang et al. [29]	IEEE IoT J., 2021	Federated Deep RL	Intrusion detection in IIoT	Edge-assisted collaborative intrusion detection
Sun et al. [30]	IEEE TNSM, 2022	Adaptive RL	IoT anomaly detection	RL-based anomaly detection for dynamic IoT
Zheng et al. [31]	IEEE TIFS, 2021	Deep RL	APT detection in CPS	Detecting advanced persistent threats in CPS
Lin et al. [32]	IEEE TWC, 2023	Multi-agent DRL	Wireless network security	Joint resource allocation and security enhancement
Liu et al. [33]	JISA, 2022	Distributed RL	DDoS defense in cloud	Privacy-preserving collaborative defense against DDoS
Mousavi et al. [34]	Computer Networks, 2023	RL-SDN	Secure routing, DoS defense	RL-based secure routing and DoS attack defense
Li et al. [35]	IEEE IoT J., 2024	Hybrid DQN + Meta-learning	Adaptive IoT security	Adaptive security via hybrid deep Q-network/meta-learning
Bai et al. [36]	IEEE TSG, 2023	Deep RL	Smart grid comm. security	Proactive secure topology reconfiguration
Hu et al. [37]	IEEE TNSE, 2023	Cooperative Deep RL	Network security risk assessment	Dynamic risk assessment using cooperative RL
Zhou et al. [38]	Future Gen. Comp. Syst., 2023	Adaptive RL	SDN security policy optimization	RL-based real-time policy optimization in SDN
Bao et al. [39]	IEEE IoT J., 2021	Multi-agent RL	Smart grid cyber-physical security	Decentralized cyber-physical defense
Gao et al. [40]	IEEE TMC, 2020	DRL, Joint Optimization	Secure resource allocation in MEC	Joint RL-based defense and resource optimization
Rahman et al. [41]	Computer Networks, 2022	CNN + RL	IoT malware detection	Hybrid CNN-RL framework for dynamic malware detection
He et al. [42]	IEEE TIFS, 2021	Deep RL (A3C)	Insider threat detection	Asynchronous RL for early insider attack detection
Wu et al. [43]	IEEE TNSE, 2023	Energy-aware RL	Secure routing in WSN	RL-based routing with energy and security constraints
Yu et al. [44]	IEEE TITS, 2023	Meta-RL	V2X cyber-attack defense	Fast adaptation to novel vehicle network attacks

work environments, learn from new threat intelligence, and adjust defensive actions on-the-fly to counteract zero-day attacks, advanced persistent threats (APTs), and rapidly mutating malware. By enabling continual self-optimization and low-latency responses, online RL-based defenses can maintain a robust security posture even in highly dynamic and unpredictable settings, such as cloud-native platforms and software-defined infrastructures.

- **Transfer Learning and Meta-Reinforcement Learning:** To address the challenges posed by rapidly changing threat landscapes and scarce labeled data, recent studies have explored the use of transfer reinforcement learning and meta-reinforcement learning (meta-RL). Transfer learning allows RL agents to leverage knowledge gained from previous domains or tasks to accelerate learning in new, related scenarios—improving sample efficiency and response times. Meta-RL further enables agents to quickly adapt to previously unseen attack patterns or novel operational contexts by learning high-level strategies that generalize across tasks. These approaches are crucial for robust security in heterogeneous, multi-domain, and continuously evolving network environments.

- **Adversarial and Hybrid Approaches:** The intersection of RL with adversarial machine learning and hybrid deep learning models has led to the development of more resilient security frameworks. Adversarial RL enables defenders to anticipate and counter sophisticated adversaries that attempt to manipulate RL policies or exploit vulnerabilities in the learning process. Additionally, hybrid architectures that combine RL with LSTM networks, graph neural networks (GNNs), or convolutional neural networks (CNNs) offer enhanced temporal, spatial, and contextual reasoning for complex tasks such as botnet detection, phishing prevention, and insider threat mitigation. These integrated models deliver robust and adaptive defense capabilities in the face of rapidly evolving and multi-faceted cyber threats.

Table III provides an overview of ten representative studies from 2019 to 2024, each demonstrating the breadth, versatility, and innovation of RL-based solutions in diverse security contexts. These works collectively highlight the evolution of RL from foundational research in generic network environments to practical, domain-specific, and highly adaptive applications. These complementary works demonstrate the ongoing evolution and broadening scope of RL research in network security. They not only address classical security problems but also pave the way for resilient and intelligent solutions in rapidly emerging and highly dynamic environments. By incorporating RL into areas such as industrial automation, blockchain, automotive safety, next-generation wireless networks, and adaptive learning, researchers are equipping modern networks with proactive, scalable, and context-aware defense mechanisms capable of meeting the demands of future digital ecosystems.

III. Protecting Networks using RL

A. DoS/DDoS attacks

In this section, we have reviewed the research in this field and the RL models that are deployed. We have categorized them based on their focused attack; A summary of them is presented in Table IV, and the taxonomy is shown in Figure 3. All blank cells in Table IV have been completed as recommended. A checkmark (✓) denotes the presence of a property or feature, while ”n/a” indicates that the property is not applicable or not explicitly reported in the original reference. This revision makes the comparative analysis of RL models for network security more clear and user-friendly for readers. To keep 5G slices isolated enough, where Denial of Service (DoS)/Distributed Denial of Service (DDoS) attacks on one slice have the lowest negative impact on the other slices, Javadpour et al., Javadpour et al. [70, 71] proposed an actor-critic RL model, called Slice Isolation based on Reinforcement Learning (SIRL). The primary goal of SIRL is to map as many requested virtual networks as possible onto the physical nodes while keeping them separate. As a result, when the adversary launches a DoS/DDoS attack against one of the physical resources, the lowest number of virtual networks go inaccessible. The reward function of this model is presented in Equation 26. When the agent in SIRL acts a, the current virtual node must be mapped on the a^{th} physical node. The states in SIRL are in the form of Equation 25. The importance value defined for each physical node in SIRL is the sum of the importance values of the links connected to that node. The importance value of a physical link is also defined as the ratio of the remaining CPU of its adjacent physical nodes to the number of physical nodes with a shared adjoining node.

Model 2: SIRL [70]

$$\mathcal{S} = \begin{bmatrix} rcpu_1 & rbwd_1 & map_1 & vld_1 & imp_1 \\ rcpu_2 & rbwd_2 & map_2 & vld_2 & imp_2 \\ \vdots & \ddots & & & \vdots \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ rcpu_N & rbwd_N & map_N & vld_N & imp_i \end{bmatrix}_{RANK} \tag{25}$$

$$\mathcal{R} = \begin{cases} -\infty, & su \neq 1 \\ \sum_{i \in PL} rbwd_i - xmap, & cu = V \ and \ su = 1 \\ \sum_{i \in PL} rbwd_i, & cu \neq V \ and \ su = 1 \end{cases} \tag{26}$$

N	:	*The total number of physical nodes*
PL	:	*The ordered set of physical links*
V	:	*The number of virtual nodes in the current request*
cu	:	*The index of the current virtual node in the current request*
su	:	*A binary value, which is one if the current virtual node is successfully mapped*
$xmap$	:	*The maximum number of virtual nodes that are mapped on a single physical node*
$rcpu_i$	:	*The remained CPU of the i^{th} physical node*
$rbwd_i$	:	*The remained bandwidth of links connected to the i^{th} physical node*
map_i	:	*The number of virtual nodes that are mapped on the i^{th} physical node*
vld_i	:	*A binary value, which is one if the current virtual node can be mapped on the i^{th} physical node*
imp_i	:	*The importance value of the i^{th} physical node*

The reward function of SIRL is improved by Javadpour et al. [71], where two conditions are considered for mapping a virtual node on the physical resources. If the mapped virtual node is the last node of the received request, SIRL is in the overall mapping situation; otherwise, it is in partial mapping. Different rewards are generated for each of these conditions, which are explained in Equation 27.

TABLE III
EMERGING RL/ML/AI APPROACHES FOR ADVANCED NETWORK SECURITY APPLICATIONS (2019–2024)

Reference	**Venue/Year**	**Technique/Model**	**Target Security Problem**	**Key Contribution**
Wu et al. [45]	IEEE TIFS, 2023	RL + Blockchain	Secure data sharing in IIoT	RL-enabled blockchain for secure collaborative sharing
Zhao et al. [46]	IEEE IoT J., 2022	Deep RL	Intrusion detection in connected vehicles	RL-based adaptive IDS for automotive networks
Hassan et al. [47]	IEEE Network, 2021	RL for 6G	Adaptive security in 6G networks	RL-driven adaptive mechanisms for 6G security
Tang et al. [48]	Computer Networks, 2022	Transfer RL	Cross-domain security in IoT	Transfer RL for leveraging knowledge across domains
Sadeghi et al. [49]	IEEE Access, 2022	Online RL	Real-time anomaly detection	Fast online RL for real-time security adaptation
Gupta et al. [50]	JISA, 2023	Deep RL	Phishing attack detection	Deep RL for automated phishing threat mitigation
Park et al. [51]	IEEE TIFS, 2021	Hybrid RL + LSTM	Botnet detection	LSTM-enhanced RL for dynamic botnet defense
Zhang et al. [52]	IEEE TII, 2022	RL-based Adaptive Control	Security in IIoT	RL-based dynamic access and threat management
Kim et al. [53]	Pattern Recognition, 2023	Adversarial RL	Robust defense under adversarial attacks	Adversarial RL for improved attack resilience
He et al. [54]	IEEE TNNLS, 2024	Meta-RL	Fast adaptation for unknown threats	Meta-RL for rapid learning under novel attacks
Ali et al. [55]	IEEE IoT J., 2022	Collaborative RL	Privacy-preserving IoT security	RL for privacy-aware collaborative defense
Luo et al. [56]	IEEE TMC, 2023	Federated RL	Anomaly detection in edge-cloud networks	Federated RL for robust anomaly response
Shen et al. [57]	Computer Networks, 2021	Adaptive RL	Dynamic attack graph analysis	RL for adaptive attack response
Meng et al. [58]	Future Gen. Comp. Syst., 2024	Hybrid RL	Malware detection in SDN	Hybrid RL for dynamic malware response
Wang et al. [59]	IEEE TIFS, 2021	Hierarchical RL	Cyber-physical attack mitigation	HRL for multi-layer security management
Garg et al. [60]	IEEE Access, 2023	Explainable AI (XAI), DNN	Network anomaly detection	XAI-based deep models for transparent anomaly detection
Liang et al. [61]	Future Gen. Comp. Syst., 2022	Multi-objective DRL	SDN threat mitigation	Multi-objective DRL for balanced threat mitigation
Ma et al. [62]	JISA, 2021	Federated RL	IoT intrusion detection	Federated RL for decentralized IoT IDS
Dong et al. [63]	Computer Networks, 2020	RL + Homomorphic Encryption	Secure communications in IoT	Homomorphic RL for privacy-preserving communication
Jin et al. [64]	IEEE TITS, 2022	Deep RL	Drone security in VANETs	DRL for secure UAV communication in vehicular networks
Deng et al. [65]	IEEE IoT J., 2023	Adversarial RL	IIoT anomaly resilience	RL-driven robust anomaly detection in IIoT
Fan et al. [66]	IEEE Access, 2020	Deep RL	Wireless intrusion prevention	DRL for adaptive wireless intrusion prevention
Cheng et al. [67]	IEEE TIFS, 2021	RL for Cyber Insurance	Security investment optimization	RL for optimizing cyber insurance in networks
Han et al. [68]	IEEE TNSM, 2022	Double DQN	SDN traffic anomaly detection	Improved DQN for real-time SDN anomaly detection
Raza et al. [69]	Computer Communications, 2024	AI-Driven RL	DDoS attack mitigation in edge	AI-RL hybrid for proactive edge DDoS defense

Model 3: SIRL [71]

$$\mathcal{R} = \begin{cases} (cdeg \times xmap)^{-1}, & cu \neq V \textit{ and } su = 1 \\ -1, & cu \neq V \textit{ and } su \neq 1 \\ (cdeg \times xmap)^{-1} + 500, & cu = V \textit{ and } \lambda_1 \\ (cdeg \times xmap)^{-1} - 500, & cu = V \textit{ and } \lambda_2 \\ -501, & cu = V \textit{ and } su \neq 1 \end{cases} \tag{27}$$

V	: *The number of virtual nodes in the current request*
cu	: *The index of the current virtual node in the current request*
su	: *A binary value, which is one if the current virtual node is successfully mapped*
$tsuc$	: *A binary value, which is one if the current request is completely mapped*
$cdeg$	: *The number of unmapped virtual links connected to the current virtual node*
$xmap$	: *The maximum number of virtual nodes that are mapped on a single physical node*
λ_1	: *The condition of having $su = 1$ and $tsu = 1$*
λ_2	: *The condition of having $su = 1$ and $tsu \neq 1$*

For securing Software-Defined Network (SDN) hosts against DDoS attack, Yungaicela-Naula et al. [72] proposed a slow-rate mitigation approach, called Deep Reinforcement Learning-based Intrusion Prevention System (DRL-IPS), that uses RL for significantly set the forwarding rules. An RL agent is considered for each connection link in the network. An Intrusion Detection System (IDS) is located in the network to classify the traffic passing through each link into normal and malicious. Once a link is identified to have flooding traffic (i.e., an alarm from the IDS is received), the agent dedicated to that link sets a blocking rule on the related network switches. However, blocking is not in the action space of the agents in their training phase. The primary focus of the training phase in DRL-IPS is to enable the agents to determine when to remove the previously set blocking rules. As a result, the action space of these agents has only two actions: removing the blocking rules and doing nothing. The environment states of DRL-IPS are in the form of Equation 28. The reward function of DRL-IPS for the agent related to the link between the i^{th} and the j^{th} nodes is also defined based on Equation 29.

Model 4: DRL-IPS [72]

$$\mathcal{S} = \left[\begin{array}{ccccc} 0 & sus_{1,2} & sus_{1,3} & \dots & sus_{1,N} \\ 0 & 0 & sus_{2,3} & \dots & sus_{2,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & sus_{N-1,N} \\ 0 & 0 & 0 & \dots & 0 \\ \hline 0 & flo_{1,2} & flo_{1,3} & \dots & flo_{1,N} \\ 0 & 0 & flo_{2,3} & \dots & flo_{2,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & flo_{N-1,N} \\ 0 & 0 & 0 & \dots & 0 \\ \hline 0 & ubwd_{1,2} & ubwd_{1,3} & \dots & ubwd_{1,N} \\ 0 & 0 & ubwd_{2,3} & \dots & ubwd_{2,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & ubwd_{N-1,N} \\ 0 & 0 & 0 & \dots & 0 \end{array}\right] \tag{28}$$

$$\mathcal{R} = \begin{cases} 1, & sus_{i,j} = 0 \textit{ and } ubwd_{i,j} > 0 \\ 0, & sus_{i,j} = 0 \textit{ and } \lambda_1 \\ 0.2, & sus_{i,j} = 0 \textit{ and } \lambda_2 \\ e^{flo_{i,j}}, & \textit{otherwise} \end{cases} \tag{29}$$

N	: *The total number of nodes*
a	: *The selected action*
$sus_{i,j}$	: *A binary value, which is one if at least one of the flows in the link between the i^{th} and the j^{th} nodes is detected as suspicious by the IDS*
$flo_{i,j}$	: *The exponential function of the number of flows detected as suspicious in the link between the i^{th} and the j^{th} nodes*
$ubwd_{i,j}$	: *The bandwidth utilization ratio of the link between the i^{th} and the j^{th} nodes*
λ_1	: *The condition of having $ubwd_{i,j} = 0$ and $a =$ "remove blocking rules"*
λ_2	: *The condition of having $ubwd_{i,j} = 0$ and $a =$ "do nothing"*

The interaction between botnet components and safe nodes in a network is modeled as a Werewolf game by Javadpour et al., Javadpour et al., Javadpour et al., Javadpour et al., Javadpour et al. [73, 74, 75, 76, 77], where an RL model, which we call Werewolf-RL, is used for detecting the launched DDoS attacks by the bots and their origins. The detection and deception components play the role of Detective and Proof, respectively; the legitimate nodes are the Townies, and the botnet components are in the group of Werewolves. The trained agent in Werewolf-RL shows Detective the optimal strategy for monitoring the Werewolves, not the Townies. The agent's goal is to sort the players in the game based on their probability of being a Werewolf. In each step, two players are compared, and the action is to change their order or not. The environment states are defined based on Equation 30. The reward function in Werewolf-RL is the number of Werewolves at the beginning of the sorted list of players generated by the Detective (i.e., agent). The exact definition of this function is presented in Equation 31. It must be noted that, since the werewolves cannot logically be detected without errors or misses, Werewolf-RL requires related datasets to train the agent.

Model 5: Werewolf-RL [73]

$$\mathcal{S} = \begin{bmatrix} in_1 & ex_1 & pr_1 & po_1 & ne_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ in_N & ex_N & pr_N & po_N & ne_N \end{bmatrix} \tag{30}$$

$$\mathcal{R} = \sum_{i=1}^{wer} \textit{det}(i) \tag{31}$$

N	: *The total number of nodes (i.e., players)*
$were$	: *The number of botnet components in the network (i.e., the number of werewolves in the game)*
in_i	: *A binary value, which is one if the i^{th} node is active (i.e., if the i^{th} player is still in the game)*
ex_i	: *The number of connections from the i^{th} node toward the legitimate nodes which are crashed by DDoS (i.e., the number of the i^{th} player's votes to the Townies who are out of the game)*
pr_i	: *The number of connections from the i^{th} node toward the deception component (i.e., the number of the i^{th} player's vote to the revealed Proof)*
po_i	: *A binary value, which is one if the i^{th} node is previously monitored by the IDS and it is identified as malicious (i.e., if the i^{th} player is previously detected by the Detective and it is identified as Werewolf)*
ne_i	: *A binary value, which is one if the i^{th} node is previously monitored by the IDS and it is identified as legitimate (i.e., if the i^{th} player is previously detected by the Detective and it is identified as Townie)*
$\text{det}(i)$	: *A binary value, which is one if the i^{th} node in the sorted list of nodes generated by the agent is truly malicious (i.e., if the i^{th} player in the sorted list of players generated by the Detective is truly Werewolf)*

B. *Epidemic attacks*

In epidemic attacks, adversaries inject malware into the network and attempt to compromise as many nodes as possible by propagating the infection. This powerful army of compromised devices is then used for their malicious purposes, such as launching a DDoS attack against a critical network asset.

Zhang et al. [78] proposed a security framework, called Deep REinforcement learning-based Vulnerability-Aware Network adaptations (DREVAN), by which the network topology is changed so that its susceptibility of enlarging infected nodes group is reduced. In contrast, the network connectivity is kept acceptable. DREVAN uses heuristic algorithms to rank the edges based on their vulnerability level in propagating infection and then suggests a pair of numbers by a Deep Reinforcement Learning (DRL) model, indicating the number of edges to be added (i.e., addition budget) and removed (i.e., removal budget). The agent in DREVAN has to add the least vulnerable edge, remove the most vulnerable one, or stop the network topology changes in each episode. The reward function of DREVAN is defined as Equation 33. The agent action is either ADD, *REMOVE*, or *STOP*, indicating that the agent wants to add an edge, remove an edge, or terminate the episode. The number of times the agent has performed *ADD* and *REMOVE* becomes the addition and removal budgets, respectively. The structure of the states in DREVAN is as Equation 32.

Model 6: DREVAN [78]

$$\mathcal{S} = \begin{bmatrix} ba & br & \begin{bmatrix} c_{1,1} & c_{1,2} & \dots & c_{1,N} \\ c_{2,1} & c_{2,2} & \dots & c_{2,N} \\ \vdots & \ddots & \vdots & \\ c_{N,1} & c_{N,2} & \dots & c_{N,N} \end{bmatrix} \end{bmatrix} \quad (32)$$

$$\mathcal{R} = \left| size_{t+1} - \frac{comps_{t+1}}{N} \right| - \left| size_t - \frac{comps_t}{N} \right| \quad (33)$$

N	:	*The total number of nodes*
ba	:	*The budget for adding edges*
br	:	*The budget for removing edges*
$c_{i,j}$	:	*A binary value, which is one if the i^{th} and the j^{th} node are connected until this step*
$size_t$	:	*The number of nodes in the largest connected subgraph of the network at time t*
$comps_t$	:	*The number of compromised nodes at time t*

Li et al. [79] proposed a detection framework, called Deep Reinforcement learning-based Worm Detection (DRWD) for detecting the spread of worms in smart grids. In a smart grid network, each end device is connected to a smart meter, and the smart meters transfer the information between the data concentrators and the end devices (i.e., smart meters). The data concentrators then forward the packets to the control center. The environment state in DRWD consists of four features as Equation 34. The action in DRWD is the worm detection performance by which the worm alarms are generated. The reward function is defined based on Equation 35. Moreover, the selected action, say a, is of size two. The first action is to classify the current flow as malicious and otherwise as legitimate.

Model 7: DRWD [79]

$$\mathcal{S} = \begin{bmatrix} \begin{bmatrix} lg_{1,t} \\ lg_{2,t} \\ \vdots \\ lg_{DS,t} \end{bmatrix} & ms_{t-1} & fl_{t-1} & infc_{t-1} \end{bmatrix} \quad (34)$$

$$\mathcal{R} = -1 \times (ms_t + fl_t + \sigma_{lat} \times lat_t) \quad (35)$$

DS	:	*The total number of data concentrators*
t	:	*The current time interval*
$lg_{i,t}$	:	*The traffic log collected by the i^{th} data concentrators in time interval t*
ms_t	:	*The ratio of the received packets that are missed to be detected as malicious to the total number of received packets until time interval t*
fl_t	:	*The ratio of the received packets that are legitimate but falsely detected as malicious until time interval t*
$infc_t$	:	*The predicted number of malicious nodes until time interval t*
lat_t	:	*The detection latency for the current packet until time interval t*
σ_{lat}	:	*The latency weight factor*

C. Authentication/Authorization attacks

The network defender can locate fake resources in the network, which pretend to be critical assets while they do not have any production values. Wang et al. [80] defined a scenario called Attacker Defender Scenario (ADS), in which the defender deploys fake resources instead of confidential ones on network hosts. Accessing the fake resources is a sign of malicious activity, and the adversary is detected. The defender in ADS uses an RL model, called ADS-RL, to effectively locate the fake resources. In other words, RL helps the defender deploy the fake traps on the network hosts, which are more probable to be accessed by the adversary. The environment state is the alarm signal that shows if the i^{th} node is compromised by the adversary. Hence, we have Equation 36. The action is locating a trap on each network node or not, and if yes, which trap is located. Since the traps are considered heterogeneous, an action is indicated with a matrix of $N \times TR$, where N is the total number of nodes, and TR is the total number of traps. Moreover, each node can have at most one trap. Therefore, we have Equation 37. The reward function is -1 if the adversary is not detected and successfully escapes after reaching the goal. If the defender detects the adversary, the reward is +1; otherwise, it is zero.

Model 8: ADS-RL [80]

$$\mathcal{S} = [alrm_1 \quad alrm_2 \quad \dots \quad alrm_N] \quad (36)$$

$$\forall \, [trp_{i,j}]_{N \times TR} \in \mathcal{A} : \sum_{j=1}^{TR} trp_{i,j} = 1 \quad (37)$$

N	:	*The total number of nodes*
TR	:	*The total number of traps*
$alrm_i$	:	*A binary value, which is one if the i^{th} node is detected as a malicious one*

$trp_{i,j}$	: *A binary value, which is one if the j^{th} trap is dedicated to the i^{th} node*

Lu et al. [81] proposed an RL-based authentication mechanism, called Deep Reinforcement Learning-based Physical layer Authentication (DRLPA), to stop rogue nodes from disturbing legitimate ones in vehicular networks. The rogue vehicles send false information, such as warning messages of unreal congestions, with the identity of other legitimate nodes. The states in DRLPA are in the form of Equation 38. The agent in DRLPA decides the authentication mode and the quantization-based threshold. The authentication mode can be fast or safe, and the safe authentication process is more complex than the fast one. The quantization-based threshold also indicates the received packet has to be identified as a spoofing signal or passed to the higher-layer protocols for further authentications. The reward function of DRLPA is defined as Equation 39.

Model 9: DRLPA [81]

$$\mathcal{S} = [fl_{t-1} \quad ms_{t-1} \quad ma_{t-1} \quad pr_t] \tag{38}$$

$$\mathcal{R} = \sigma_{cs} \times cs - ma_t \times pr_t \times (\sigma_{fl} \times fl_t + \sigma_{ms} \times ms_t) \tag{39}$$

t	: *The current time interval*
cs	: *The authentication cost*
fl_t	: *The ratio of the received packets falsely detected as malicious to the total number of received packets until time interval t*
ms_t	: *The ratio of the received packets that are missed to be detected as malicious to the total number of received packets until time interval t*
ma_t	: *The ratio of the total received malicious packets to the total number of received packets until time interval t*
pr_t	: *The priority level of the new received packet in time interval t*
σ_{fl}	: *The false detection rate factor*
σ_{ms}	: *The missed detection rate factor*
σ_{cs}	: *The authentication cost factor*

Another security model for preventing malicious nodes in vehicular networks from performing authenticity attacks is proposed by Liu et al. [82] and called Deep Reinforcement Learning-based Intelligent Message Forwarding (DRL-IMF). This model securely forwards the messages from a source node to the destination, passing through middle nodes. DRL-IMF avoids the suspicious middle nodes as much as possible while keeping the messages' urgency and delay constraints in mind. The environment states are in the form of Equation 40. In the current state, which shows the features of the current node in the forwarding path, the agent action is interpreted as the next hop for forwarding the message. If the agent's action is 0, the message is no longer forwarded. In other words, the agent thinks that the current node is the destination. Otherwise, a number, say a, between 1 to the total number of neighbor nodes of the current node is selected, which means the message has to be forwarded to the a^{th} neighbor node. The reward function in DRL-IMF is defined as Equation 41.

Model 10: DRL-IMF [82]

$$\mathcal{S} = \begin{bmatrix} \begin{bmatrix} del_1 \\ \vdots \\ del_{NN} \end{bmatrix} & \begin{bmatrix} dist_1 \\ \vdots \\ dist_{NN} \end{bmatrix} & bdst & urg & reli & bml \end{bmatrix} \tag{40}$$

$$\mathcal{R} = \begin{cases} 0, & \lambda_{bdst} \\ \dfrac{-1}{sig(reli)} + \beta_{othr}, & \lambda_{bml} \\ \dfrac{-\sigma_{del} \times del_a}{sig(urg)} - (1-\sigma_{del}) \times dist_a, & \lambda_{othr} \end{cases} \tag{41}$$

NN	: *The maximum number of neighbor nodes*
a	: *The selected neighbor node*
$bdst$	: *A binary value, which is one if the current node is the destination node*
urg	: *A number between 1 and 30 indicating the urgency of the forwarded message*
$reli$	: *A number between 1 and 30 indicating the reliability level the forwarded message requires*
bml	: *A binary value, which is one if the current node is malicious*
del_i	: *The delay of forwarding the message to the i^{th} neighbor node*
$dist_i$	: *The distance between the destination node and the i^{th} neighbor node*
σ_{del}	: *The weight factor of delay*
λ_{bdst}	: *The condition, in which the selected neighbor node is the destination*
λ_{bml}	: *The condition, in which the selected neighbor node is malicious*
λ_{othr}	: *The condition, which is neither λ_{bdst} nor λ_{bmal}*
β_{othr}	: *The output of reward function in the condition of having λ_{othr}*
sig(x)	: *The denominator of sigmoid function with the input of x, which is equal to $1 + e^{-(x-10)}$*

To secure the process of offloading users' tasks to the cloud computing nodes against unauthorized devices, Zhang et al. [83] proposed an RL model, called Secure Resource Allocation based on Deep Reinforcement Learning (SRA-DRL), that reduces the energy consumption cost, while keeping the delay and the security constraints satisfied. When a task arrives, the computing node checks the delay constraint and then, performs the most secure encryption algorithm, the execution time of which does not exceed the constraint. The states in SRA-DRL are defined as the user's request, the detail of which is shown in Equation 42. Each action can be specified as a binary array, say $\{a_0, a_1, \ldots, a_N\}$, where N is the total number of cloud computing nodes, and a_i indicates whether or not the i^{th} task is offloaded to the i^{th} node. The i^{th} node is the edge computing node receiving the user's task and the i^{th} cloud computing node for $i = 0$ and $i > 0$, respectively. Moreover, the reward function in SRA-DRL (Equation 43) is defined as the negative of the cost, which is the summation of delay cost and energy consumption cost.

Model 11: SRA-DRL [83]

$$\mathcal{S} = \begin{bmatrix} dsize_1 & rcpu_1 & delc_1 \\ \vdots & \vdots & \vdots \\ dsize_{AT} & rcpu_{AT} & delc_{AT} \end{bmatrix} \tag{42}$$

$$\mathcal{R} = -(\sigma_{del} \sum_{i=1}^{AT} del_i + \sigma_{enrg} \sum_{i=1}^{AT} enrg_i) \tag{43}$$

AT	:	*The number of arrived tasks*
$dsize_i$	:	*The data size of the i^{th} task*
$rcpu_i$	:	*The required CPU of the i^{th} task*
$delc_i$	:	*The delay constraint of the i^{th} task*
del_i	:	*The delay of the i^{th} task*
$enrg_i$	:	*The consumed energy for processing the i^{th} task*
σ_{del}	:	*The weight factor of delay in the total cost*
σ_{enrg}	:	*The weight factor of consumed energy in the total cost*

A similar approach to SRA-DRL's for protecting mobile tasks during the offloading process to edge or cloud computing nodes is proposed by Nguyen et al. [84], which is called Advanced Deep Reinforcement Learning -based Offloading algorithm (ADRLO). The authorized mobile devices are identified using smart contracts on blockchain in ADRLO. The cost of related processes to the smart contract (e.g., verification) and offloading costs are minimized. The key difference between SRA-DRL and ADRLO is that SRA-DRL assumes multiple cloud nodes, while ADRLO consists of a single cloud component. In the latter, tasks meant for the cloud are first sent to the edge layer before being forwarded to the cloud node. The environment states in ADRLO are triples of the total offloading cost, the available resources on the edge/cloud server, and its available bandwidth. The actions are defined as an array in the form of Equation 44. The reward function is similar to that of SRA-DRL, where the negative of the cost is considered. The details of the reward function in ADRLO are presented in Equation 45.

Model 12: ADRLO [84]

$$\mathcal{A} = \begin{bmatrix} ed\_alc_1 & cl\_alc_1 & rsrc\_alc_1 & bdwd\_alc_1 \\ \vdots & \vdots & \vdots & \vdots \\ ed\_alc_N & cl\_alc_N & rsrc\_alc_N & bdwd\_alc_N \end{bmatrix} \tag{44}$$

$$\mathcal{R} = -(off\_cst + cnrt\_cst) \tag{45}$$

N	:	*The total number of mobile devices*
off_cst	:	*The offloading cost*
$cnrt\_cst$	:	*The contract cost*
ed_alc_i	:	*A binary value, which is one if the task related to the i^{th} device is offloaded to the edge node*
cl_alc_i	:	*A binary value, which is one if the task related to the i^{th} device is offloaded to the cloud node*
$rsrc\_alc_i$	:	*The allocated resources units to the task related to the i^{th} device*
$bdwd\_alc_i$	:	*The allocated bandwidth units to the task related to the i^{th} device*

D. General attacks

Jiang et al. [85] proposed a DRL agent, called DRL VNE algorithm based on differentiated Quality of service and Security requirements (QS-DRL-VNE), that maps the virtual node requests on the physical resources, considering their security level. Users who request virtual networks accept a specific threshold of security level. This requirement is sent to QS-DRL-VNE, and the requested network is mapped on the physical resources with a security level equal to or higher than the requested one. QS-DRL-VNE defines the revenue consumption ratio as its reward function, which is the ratio of the sum of virtual nodes' CPU and virtual links' bandwidth of the successfully mapped request to the sum of CPU and bandwidth it has occupied. To be more precise, one can calculate the reward function in QS-DRL-VNE as Equation 47. The actions are defined as the physical nodes on which the current virtual node can be mapped. The states of the problem environment in QS-DRL-VNE are in the form of Equation 46.

Model 13: QS-DRL-VNE [85]

$$\mathcal{S} = \begin{bmatrix} rcpu_1 & bwd_1 & del_1 & sec_1 \\ rcpu_2 & bwd_2 & del_2 & sec_2 \\ \vdots & \vdots & \vdots & \vdots \\ rcpu_N & bwd_N & del_N & sec_N \end{bmatrix} \tag{46}$$

$$\mathcal{R} = \frac{\sum_{i \in VN} vcpu_i + \sum_{i \in L} vbwd_i}{\sum_{i \in VN} vcpu_i + \sum_{i \in L} vbwd_i \times hp_i} \tag{47}$$

N	:	*The total number of physical nodes*
V	:	*The set of virtual nodes in the current request*
L	:	*The set of virtual links in the current request*
$rcpu_i$	:	*The remained CPU of the i^{th} physical node*
$vcpu_i$	:	*The required CPU of the i^{th} virtual node*
bwd_i	:	*The sum of bandwidth of the links connected to the i^{th} physical node*
$vbwd_i$	:	*The required bandwidth of the i^{th} virtual link*
del_i	:	*The delay constraint of the i^{th} physical node*
sec_i	:	*The security level of the i^{th} physical node*
hp_i	:	*The number of physical links (i.e., hops) the i^{th} virtual link is mapped on*

A similar agent to QS-DRL-VNE, called Security perception Algorithm for VNE based on RL (SA-RL-VNE), is proposed by Jiang et al. [86], and the difference between these two agents is in the features they use to present the environment. The considered features for the physical nodes are presented in Equation 48.

Model 14: SA-RL-VNE [86]

$$\mathcal{S} = \begin{bmatrix} rcpu_1 & deg_1 & dist_1 & sec_1 \\ rcpu_2 & deg_2 & dist_2 & sec_2 \\ \vdots & \vdots & \vdots & \vdots \\ rcpu_N & deg_N & dist_N & sec_N \end{bmatrix} \tag{48}$$

N	:	*The total number of physical nodes*
$rcpu_i$	:	*The remained CPU of the i^{th} physical node*
deg_i	:	*The total number of links connected to the i^{th} physical node*
$dist_i$	:	*The average distance between the i^{th} and the other physical nodes*
sec_i	:	*The security level of the i^{th} physical node*

For mapping Internet of Things (IoT) devices' jobs on virtual machines, considering their security requirement, another RL model, called Deep Reinforcement Learning for optimization in Software-Defined Security (DRL-SDS), is proposed by Li and Qin [87]. Each arrived job requires a specific quality of service level, and the security constraints are also included in it. In other words, the amount of acceptable delay indicates the complexity of the security approach. On the other hand, each of the virtual resources has specific features, such as renting or start-up cost and computing or input/output speed. The jobs are assigned to the resources that can satisfy their requirements. The actions in DRL-SDS vary from 1 to N, where N is the total number of virtual resources. The actions are defined as the selected resource to handle the arrived job. The structure of environment states are as Equation 49. The reward function in this model is also defined based on Equation 50.

Model 15: DRL-SDS [87]

$$\mathcal{S} = \begin{bmatrix} \begin{bmatrix} rtyp_1 & \dots & rtyp_N \\ cspd_1 & \dots & cspd_N \\ ispd_1 & \dots & ispd_N \\ rcst_1 & \dots & rcst_N \\ scst_1 & \dots & scst_N \end{bmatrix} & jtyp & jlen & jdl \end{bmatrix} \tag{49}$$

$$\mathcal{R} = \begin{cases} 0, & jdl < wdl + edl \\ \dfrac{2 \times edl \times tan^{-1}(cst - xcst)}{-\pi \times (wdl + edl)}, & jdl \geqslant wdl + edl \end{cases} \tag{50}$$

N	:	*The total number of resource nodes*
$jtyp$	:	*The type of the arrived job*
$jlen$	:	*The length of the arrived job*
jdl	:	*The requirements of the arrived job in terms of acceptable delay*
wdl	:	*The waiting delay after the arrived job is assigned to the resources*
edl	:	*The execution delay of the arrived job*
$rtyp_i$	:	*The type of the i^{th} resource node*
$cspd_i$	:	*The computation speed of the i^{th} resource node*
$ispd_i$	:	*The input/output speed of the i^{th} resource node*
cst	:	*The total cost of handling the arrived job*
$xcst$	:	*The maximum cost a job can result*
$rcst_i$	:	*The renting or using cost of the i^{th} resource node*
$scst_i$	:	*The start-up cost of the i^{th} resource node*

An anomaly detection framework, called SparseIDS, is proposed by Bachl et al. [88], in which RL is utilized for optimally sampling the network packets. IDSs capture the traffic packets and decide whether or not they are malicious. As more packets are captured and classified, the delay of IDSs in responding to threats increases. Therefore, SparseIDS samples the packets to reduce the computational complexity and the process's duration. To capture the packets with the most relevant information, SparseIDS utilises an actor-critic RL model, where the agent receives higher rewards when a higher number of packets are skipped (sparsity reward) and achieves a higher value of classification accuracy (classification reward). The exact reward function of SparseIDS, equal to the summation of sparsity reward and classification reward with a trade-off factor, is shown in Equation 52. The agent's action in this model is defined as the number of packets between the current and the next captured ones. For example, if the action equals three, the next two packets are skipped, and the third one is inspected. The environment states in SparseIDS are in the form of Equation 51.

Model 16: SparseIDS [88]

$$\mathcal{S} = [src \quad dst \quad prot \quad plen \quad arr \quad dir \quad tcp \quad skp] \tag{51}$$

$$\mathcal{R} = \frac{\sum_{i=n+1}^{N} 1 - \mid l_i - c_i \mid}{N - n} + \sigma_{sp\_cl} \frac{\sum_{i=n+1}^{N} s_i}{N - n - 1} \tag{52}$$

N	:	*The total number of packets in the current flow*
n	:	*The total number of currently processed packets*
src	:	*The packet source port*
dst	:	*The packet destination port*
$prot$	:	*The protocol identifier*
$plen$	:	*The packet length*
arr	:	*The interarrival time to the previous packet*
dir	:	*The packet direction*
tcp	:	*The packet TCP flags*
skp	:	*The number of skipped packets*
l_i	:	*A binary value, which is one if the i^{th} packet is malicious*
c_i	:	*A binary value, which is one if the i^{th} packet is classified as malicious*
s_i	:	*A binary value, which is one if the i^{th} packet is skipped*

σ_{sp_cl}	: *A parameter indicating the trade-off between sparsity and classification*

Another intrusion detection approach based on RL is proposed by Moudoud and Cherkaoui [89], where the agent is trained with a labeled dataset and tries to predict whether or not the network traffic is malicious. The proposed model is called Intrusion Detection based on Deep Reinforcement Learning (ID-DRL), and its states are the rows of NSL-KDD, and the state size is equal to the number of features in this dataset. There are two defined actions and rewards in ID-DRL. If the agent correctly identifies the attacks, a reward of +1 is obtained; otherwise, its gained reward is -1.

To reveal general adversaries in cloud environments, Li et al. [90] have proposed a deception framework, which we refer to as Optimal Deception Framework with Deep Reinforcement Learning (ODF-DRL), that utilises RL to locate decoys in the network optimally. Deploying these decoys confuses the adversaries about which microservices are real and which ones are fake. However, implementing deceptive services has its own cost, which is why RL is used. The optimal placement of decoys may lead to the highest possible deception in the network while the cost of this defense is kept low. The action space in ODF-DRL defines whether or not to put a decoy on a node. The computing nodes construct the states as Equation 53. The reward function in ODF-DRL is also defined as Equation 54.

Model 17: ODF-DRL[90]

$$\mathcal{S} = \begin{bmatrix} expd_{1,1} & \dots & expd_{1,MS} \\ expd_{2,1} & \dots & expd_{2,MS} \\ \vdots & \ddots & \vdots \\ expd_{N,1} & \dots & expd_{N,MS} \end{bmatrix} \tag{53}$$

$$\mathcal{R} = \frac{fk\_atkp + cn\_atkp}{atkp} \tag{54}$$

N	: *The total number of nodes*
MS	: *The total number of microservices*
$atkp$	: *The total number of attack paths the adversary can select to reach its target*
fk_atkp	: *The number of fake attack paths generated by the decoys*
cn_atkp	: *The number of concealed attack paths*
$expd_{i,j}$	: *The difficulty of exploiting the j^{th} microservice on the i^{th} node*

In addition to secure mapping and offloading, as well as intrusion detection approaches, there is also research on secure routing. Mudgerikar and Bertino [91] proposed an RL model, called Smart Traffic Engineering in Software-Defined Networks (STE-SDN), that securely routes the network traffic in SDN. Some security components in the network provide security services, such as intrusion detection or prevention. Whenever network flows are forwarded through these components, they can be checked and the required protection actions can be performed. An ideal routing rule forwards a network flow through all types of security components to apply the most possible protection considerations. At the same time, the networking quality constraints, such as delay and congestion, are satisfied, and this is the main aim of STE-SDN. CP candidate paths are suggested for each flow, and the agent selects one to pass the current flow through it. These candidates are the first CP shortest path from the source node to the destination. The defined environment states in this model are in the form of Equation 55. In the implementation phase of STE-SDN, each flow is described with 80 features ($FF = 80$). The reward function of STE-SDN is also shown in Equation 56.

Model 18: STE-SDN [91]

$$\mathcal{S} = \begin{bmatrix} \begin{bmatrix} flw_1 \\ flw_2 \\ \vdots \\ flw_{FF} \end{bmatrix} & \begin{bmatrix} pcst_1 \\ pcst_2 \\ \vdots \\ pcst_{CP} \end{bmatrix} & src & dst \end{bmatrix} \tag{55}$$

$$\mathcal{R} = (1 - \sigma_s) \times del + \sigma_s \sum_{i=1}^{SC} (se_i(\sigma_p \times pr_i + \sigma_c \times cn_i)) \tag{56}$$

SC	: *The total number of security components in the network*
FF	: *the number of features considered for describing the flows*
CP	: *The number of candidate paths considered for each flow*
src	: *The source IP address of the current flow*
dst	: *The destination IP address of the current flow*
del	: *The time required for processing the current flow, or in other words, the cause of delay*
flw_i	: *The i^{th} feature value of the current flow*
$pcst_i$	: *The cost of routing the current flow through the i^{th} candidate path in terms of bandwidth*
se_i	: *A binary value, which is one if the current flow results in generating an alert by the i^{th} security component*
pr_i	: *The priority level of the alert generated by the i^{th} security component*
cn_i	: *The confidence level of the alert generated by the i^{th} security component*
σ_s	: *The security weight factor over the networking quality*
σ_p	: *The priority weight factor, which indicates how important is the priority level of the alerts generated by security components*
σ_c	: *The confidence weight factor, that indicates how important is the confidence level of the alerts generated by security components*

TABLE IV
A SUMMARY OF THE RL MODELS CATEGORIZED BASED ON THE FOCUSED ATTACKS, WHERE VB, PB, ON, OF, AND DL ARE FOR VALUE-BASED, POLICY-BASED, ON-POLICY, OFF-POLICY, AND DEEP LEARNING, RESPECTIVELY.

Model name	**Focused Attack**	**State size**	**Size of $\mathcal{A}$**	**Description**	**Model Type**				
					VB	**PB**	**ON**	**Of**	**DL**
SIRL [70]	DoS/DDoS attacks	$5N$	N	Model 2	✓	✓	n/a	✓	✓
SIRL [71]		$5N$	N	Model 2	✓	✓	n/a	✓	✓
DRL-IPS [72]		$3N^2$	2	Model 4	✓	n/a	n/a	✓	✓
Werewolf-RL [73]		$5N$	2	Model 5	✓	✓	n/a	✓	✓
DREVAN [78]	Epidemic attacks	$N^2 + 2$	3	Model 6	✓	n/a	n/a	✓	✓
DRWD [79]		$DS + 3$	2	Model 7	✓	n/a	n/a	✓	✓
ADS-RL [80]	Auth. attacks	N	$N \times TR$	Model 8	✓	n/a	n/a	✓	n/a
DRLPA [81]		4	$2QL + 2$	Model 9	✓	n/a	n/a	✓	✓
DRL-IMF [82]		$2NN + 4$	$NN + 1$	Model 10	✓	n/a	n/a	✓	✓
SRA-DRL [83]		$3OT$	$CN + 1$	Model 11	✓	n/a	n/a	✓	✓
ADRLO [84]		3	$4N$	Model 12	✓	n/a	n/a	✓	✓
QS-DRL-VNE [85]	General attacks	$4N$	N	Model 13	n/a	✓	n/a	✓	✓
SA-RL-VNE[86]		$4N$	N	Model 14	n/a	✓	n/a	✓	✓
DRL-SDS [87]		$5N + 3$	N	Model 15	✓	n/a	n/a	✓	✓
SparseIDS [88]		8	∞	Model 16	✓	✓	n/a	✓	✓
ID-DRL [89]		DF	2	-	✓	n/a	n/a	✓	✓
ODF-DRL [90]		$N \times MS$	N	Model 17	n/a	✓	n/a	✓	✓
STE-SDN [91]		$FF + CP + 2$	CP	Model 18	✓	n/a	n/a	✓	✓

$N =$ the number of network nodes	$DS =$ the number of datacenters
$TR =$ the number of deceptive traps	$QL =$ the quantization level
$NN =$ the maximum number of neighbor nodes	$CN =$ the number of computing nodes
$OT =$ the number of offloaded tasks	$DF =$ the number of dataset features
$MS =$ the number of microservices	
$FF =$ the number of features considered for describing network flows	
$CP =$ the number of candidate paths for routing the network flows	

Note: "✓" indicates the presence of the property or feature; "n/a" means not applicable or not reported in the referenced work. Model type abbreviations: VB = Value-Based, PB = Policy-Based, ON = On-policy, Of = Off-policy, DL = Deep Learning.

IV. Improving RL Models for Network Security

In this section, we present recommendations for improving RL models to enhance their applicability for network protection. These suggestions are classified into performance-, scalability-, and security-based ones, which are explained in the remainder of this section.

A. Performance

The problem presentation directly affects the performance of an RL model. If the agent is provided with a state that lacks sufficient information about the environmental situation, or the received reward is insufficient to guide the agent toward the optimal solution, even the most potent agent cannot suggest the optimal solution because it is given misleading information. Hence, the designers of the RL model must avoid weak state structures or reward functions.

1) Weak state structure: The agent explores the environment, but what happens if this environment is presented in a confusing way or even with significant losses? Recalling Scenario 1, and the Δ_{Naive} model generated and trained in Algorithm 10, assume that we have another RL model, called $\Delta_{WeakState}$, that has weak state structure as Equation 57 instead of Equation 23.

Model 19: $\Delta_{WeakState}$

$$\mathcal{S} = [sec_1 \quad sec_2 \quad \dots \quad sec_N] \tag{57}$$

N : The total number of physical resources
sec_i : The supported security level of the i^{th} physical node

In this situation, the agent is not provided with information about the mapping constraints, such as the resource limitations for hosting virtual nodes. This loss of information significantly affects the learning performance of $\Delta_{WeakState}$. It must be mentioned that we will discuss in subsection IV-B the trade-off between the accuracy of presenting the environment and the model's scalability. However, ignoring important aspects of the environment, such as those omitted by $\Delta_{WeakState}$, affects the model's performance and is not recommended.

2) Weak reward function: Another key point in designing an efficient RL model is its reward function definition. The reward function must reflect the right direction toward the optimal solution. Moreover, it has to lead the agent as much as possible. We define another weak RL model, called $\Delta_{WeakReward}$, the reward function of which is defined as Equation 58 instead of Equation 24.

Model 20: $\Delta_{WeakReward}$

$$\mathcal{R} = \begin{cases} 0, & \nexists i \in \mathbb{N}_N : vd_i = 1 \\ -1, & vd_a = 0 \textit{ and } \exists i \in \mathbb{N}_N : vd_i = 1 \\ 1, & vd_a = 1 \end{cases} \tag{58}$$

N : The total number of physical resources
$\mathbb{N}_N$: The set of natural numbers lower than $N + 1$
a : The selected action
vd_i : A binary value, which is one if the current virtual node can be mapped on the i^{th} physical node

Based on this reward function, the agent is led toward satisfying the constraints. However, as the reward is one in all the successful mapping situations, the agent does not

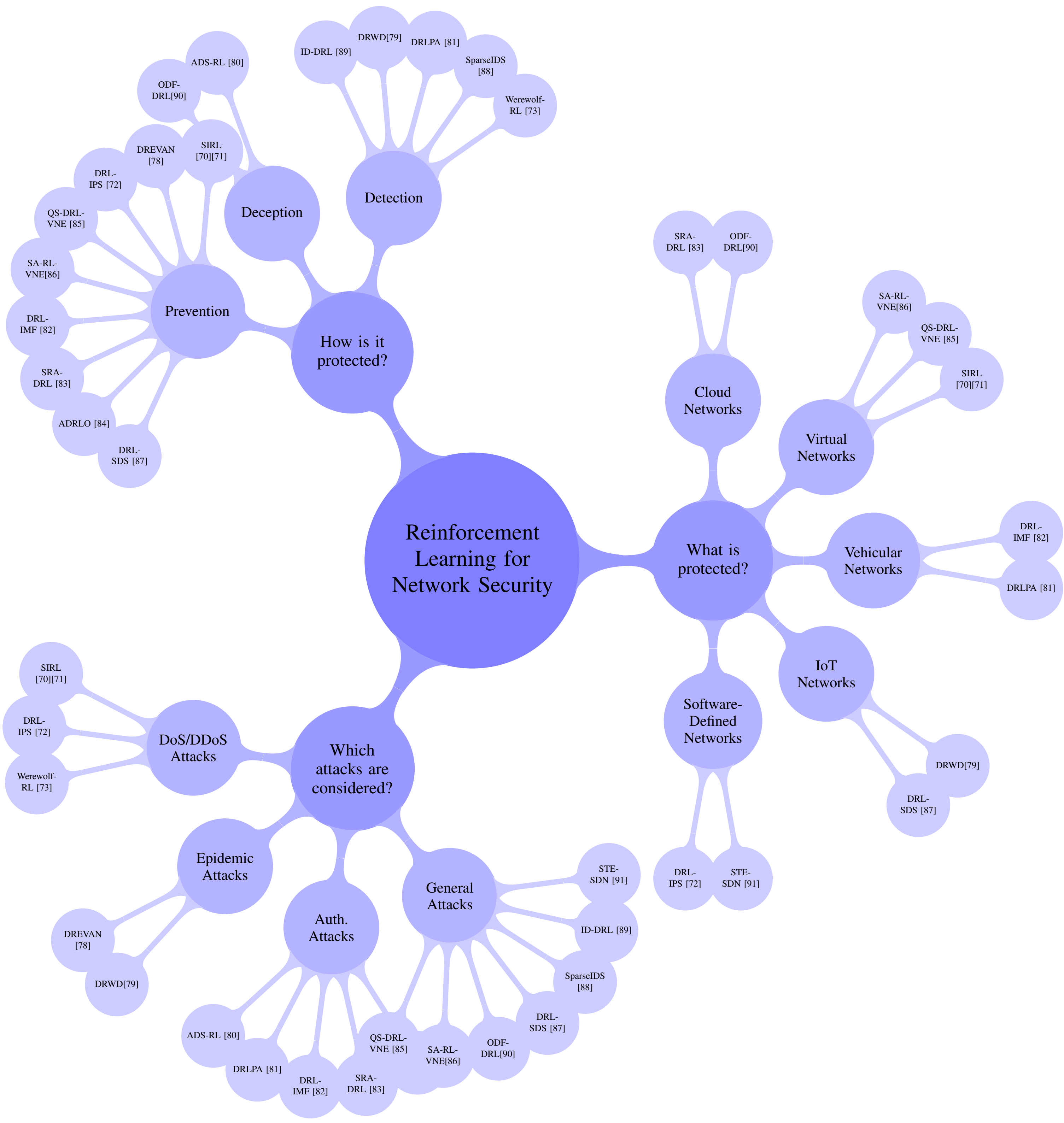


Fig. 3. The taxonomy of RL in network security

learn that assigning a request to the resource with the lowest supported security level is preferable. Therefore, the agent cannot understand the impact of its actions completely.

B. Scalability

Making an RL model scalable enough to be used in environments with different configurations is more applicable than the cases in which the model can find the optimal solution for just a few situations. In other words, it is desirable for an RL model to be trained once and then used for multiple scenarios, especially large-scale ones. Scalability helps the model secure large-scale networks, which are growing even more. Effectively defining the environment states and the action space can directly affect the model's scalability. We define another scenario, Scenario 2, with a larger scale than that of Scenario 1.

C. Implementation Resources and Best Practices

To maximize the practical utility of reinforcement learning (RL) in network security, a combination of robust frameworks, realistic simulation environments, and well-established datasets is essential. This section presents a comprehensive set of resources and practical recommendations to guide researchers and practitioners in effective RL-based cybersecurity experimentation and deployment.

1) **RL Frameworks and Libraries:**
   - *Stable-Baselines3* [92]: A widely used, user-friendly library supporting major RL algorithms (DQN, PPO, A2C, etc.) with Python and PyTorch.
   - *RLlib* [93]: A scalable, industry-grade library for distributed RL experiments.
   - *OpenAI Gym* [94]: Standardized RL environment API, enabling rapid prototyping and benchmarking.
2) **Network Simulation and Emulation Environments:**
   - *Mininet* [95]: Lightweight, flexible emulator for SDN and network experiments.
   - *NS-3* [96]: Widely adopted discrete-event network simulator for research on wireless, IoT, and Internet-scale protocols.
   - *Omnet++* [97]: Modular simulation platform for communication networks, including IoT and CPS.
3) **Public Datasets for Security Research:**
   - *CICIDS2017* [98]: Comprehensive intrusion detection dataset with multiple attack types and realistic traffic profiles.
   - *NSL-KDD* [99]: Standard benchmark for anomaly and intrusion detection in network environments.
   - *UNSW-NB15* [100]: Modern dataset including nine attack families and updated network characteristics.
4) **Best Practices for RL-based Security Experimentation:**
   - Clearly define state and action spaces tailored to the security scenario and underlying network architecture.
   - Engineer reward functions that precisely capture security objectives (e.g., attack mitigation, detection accuracy) while minimizing unintended behavior.
   - Leverage cross-validation, early stopping, and multiple random seeds to ensure robust evaluation and avoid overfitting.
   - Reproduce experiments using public datasets or open-source simulators; document all settings for full reproducibility.

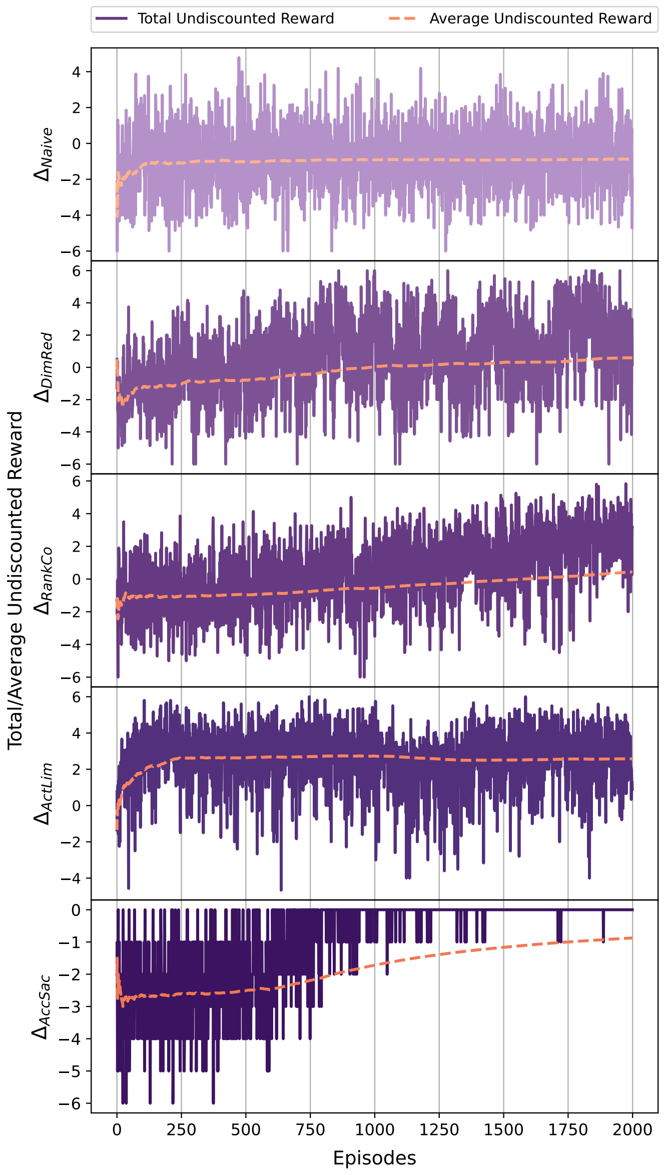


Fig. 4. Comparing the undiscounted reward of the RL models suggested for Scenario 2.

   - Report metrics such as true positive rate, false positive rate, and computational cost, enabling fair comparison with baseline methods.

By leveraging these resources and following best practices, both researchers and practitioners can more effectively translate RL theory into deployable solutions for real-world network security challenges.

Scenario 2. *The resource provider described in Scenario 1 has improved its services by increasing the maximum number of security levels the physical resources can support from 10 to 500. In other words, we have* $xsec = 500$.

Based on this scenario, we can evaluate the scalability of different RL models. The learning performance and evaluation results are shown in Figure 4 andFigure 5, respectively. As shown in Figure 5, we can see the number of accepted requests in Δ_{Naive} decrease as the scale of the problem grows. And

TABLE V
RECOMMENDED RESOURCES FOR RL-BASED NETWORK SECURITY EXPERIMENTATION

Category	Name	Reference / URL	Description
RL Framework	Stable-Baselines3	https://stable-baselines3.readthedocs.io/ [92]	PyTorch-based library for state-of-the-art RL algorithms; easy to use and well documented.
RL Framework	RLlib	https://docs.ray.io/en/latest/rllib/ [93]	Scalable RL library for distributed training and large environments.
RL Framework	OpenAI Gym	https://gym.openai.com/ [94]	Standard toolkit for developing and comparing RL algorithms.
RL Framework	CleanRL	https://github.com/vwxyzjn/cleanrl [101]	Minimal and reliable RL implementations for reproducible research.
RL Framework	Keras-RL	https://github.com/keras-rl/keras-rl [102]	Deep RL library integrated with Keras for fast prototyping.
RL Framework	Tensorforce	https://github.com/tensorforce/tensorforce [103]	Modular RL library built on TensorFlow for applied research.
RL Library	Acme (DeepMind)	https://github.com/deepmind/acme [104]	Modular and scalable framework for advanced RL research (by DeepMind).
RL Library	Dopamine (Google)	https://github.com/google/dopamine [105]	Lightweight, research-focused RL framework by Google Research.
RL Library	Spinning Up (OpenAI)	https://spinningup.openai.com/en/latest/ [106]	Educational RL resource with implementations and tutorials.
RL Environment	PettingZoo (Multi-Agent RL)	https://www.pettingzoo.ml/ [107]	Standardized environments for multi-agent RL benchmarking.
Simulator	Mininet	http://mininet.org/ [95]	Lightweight network emulator for SDN and network experiments.
Simulator	NS-3	https://www.nsnam.org/ [96]	Discrete-event network simulator for Internet systems.
Simulator	Omnet++	https://omnetpp.org/ [97]	Versatile discrete event simulator for networks and protocols.
Simulator	GNS3	https://www.gns3.com/ [108]	Graphical network simulator for complex topologies.
Simulator	CORE (Common Open Research Emulator)	https://www.nrl.navy.mil/itd/ncs/products/core [109]	Real-time network emulator for research and education.
Simulator	CyberBattleSim (Microsoft)	https://github.com/microsoft/CyberBattleSim [110]	Cybersecurity simulation environment for RL-based defense strategies.
Dataset	CICIDS2017	https://www.unb.ca/cic/datasets/malmem-2022.html [98]	Labeled dataset for network intrusion detection research.
Dataset	NSL-KDD	https://www.unb.ca/cic/datasets/nsl.html [99]	Improved version of KDD Cup 1999, widely used for IDS research.
Dataset	UNSW-NB15	https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/ [100]	Realistic network traffic dataset for evaluating security solutions.
Dataset	TON_IoT	https://research.unsw.edu.au/projects/toniot-datasets [111]	Comprehensive dataset for IoT, telemetry, and network flow analysis.
Dataset	Bot-IoT	https://research.unsw.edu.au/projects/bot-iot-dataset [112]	IoT botnet dataset for attack and anomaly detection.
Dataset	DARPA Intrusion Detection Evaluation	https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset [113]	Classic intrusion detection benchmark dataset.
Dataset	KDD Cup 1999	http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html [114]	Historical large-scale IDS dataset; foundation for IDS research.
Dataset	ISCXIDS2012	https://www.unb.ca/cic/datasets/ids.html [115]	Diverse dataset for evaluating intrusion detection systems.
Platform	OpenAI Safety Gym	https://github.com/openai/safety-gym [116]	Environments for research on safe RL and constrained control.
Platform	OpenAI Multi-Agent Particle Environments	https://github.com/openai/multiagent-particle-envs [117]	Lightweight multi-agent RL benchmarking environments.
Tool	Wireshark	https://www.wireshark.org/ [118]	Popular network protocol analyzer for packet inspection.
Tool	SDN OpenFlow Lab	https://www.opennetworking.org/sdn-openflow-lab/ [119]	Hands-on SDN/OpenFlow experimentation platform.
Benchmark	RLBench (Robot Learning)	https://github.com/stepjam/RLBench [120]	Suite for benchmarking RL algorithms in robot learning tasks.

the worst point is that its performance is even lower than a random mapping scheme. This degradation shows that Δ_{Naive} is not scalable enough for use in Scenario 2.

1) Dimension reduction: The dimension reduction technique reduces the number of input variables that construct the environment state. If the states' dimensions are reduced so that the environment is presented in fewer states, the agent learns faster and more efficiently.

Considering Scenario 2 and the inefficiency of Δ_{Naive} to find the best mapping solution, we suggest another RL model, called Δ_{DimRed}, that uses this technique to handle our desired scalability. In this model, instead of storing a pair for each resource, that indicates its security level and its capability of hosting the current virtual node (Equation 23), we can store a single number. This number is zero if the resource is invalid; otherwise, it shows the security level. Therefore, we can define

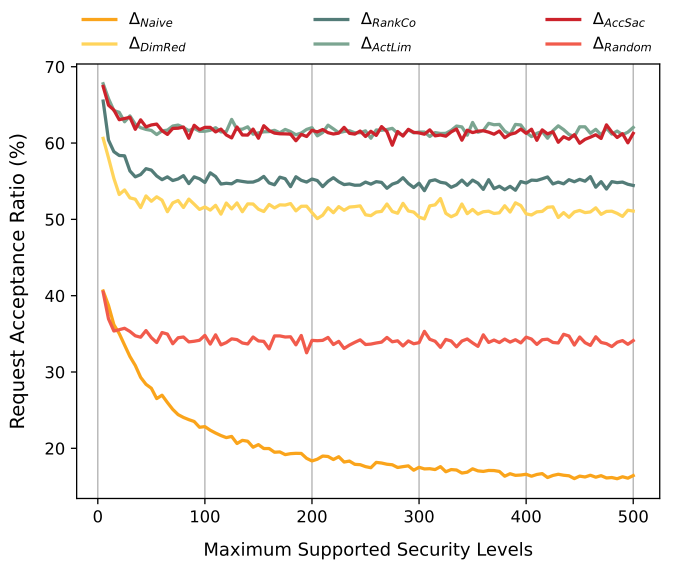


Fig. 5. Evaluating the scalability of different RL models suggested for Scenario 2 in terms of their acceptance ratio.

the state structure of Δ_{DimRed} as Equation 59.

Model 21: Δ_{DimRed}

$$\mathcal{S} = [vdsec_1 \quad vdsec_2 \quad \dots \quad vdsec_N] \tag{59}$$

N	:	*The total number of physical resources*
$vdsec_i$	:	*A number that is zero if the i^{th} resource is not capable of hosting the current request, and otherwise specifies the maximum security level the i^{th} resource can support*

Reducing the dimensions of Δ_{Naive} does not always result in better scalability, and one has to show that the number of possible states are also reduced. Now, we show that the number of possible states in Δ_{DimRed} ($|\mathcal{S}_{DimRed}|$) is less than that of Δ_{Naive} ($|\mathcal{S}_{Naive}|$), by presenting Theorem 1 and proving it.

Theorem 1. *The number of states the agent may face in Δ_{DimRed} is less than that of Δ_{Naive}. In other words, we have $|\mathcal{S}_{DimRed}| < |\mathcal{S}_{Naive}|$.*

Proof. The states in Δ_{Naive} contain N pairs, and each pair has $xsec \times 2$ possible values. Hence, the agent may face $(xsec \times 2)^N$ different states in Δ_{DimRed}. On the other hand, the states in Δ_{DimRed} contain N numbers that vary from zero to $xsec$. As a result, there are $(xsec + 1)^N$ different states in the observation set of Δ_{DimRed}. Moreover, we know that the maximum number of supported security levels is more than one. So, we conclude Equation 60.

$$xsec > 1 \Rightarrow \begin{cases} xsec + xsec > xsec + 1 \\ xsec + 1 > 2 \end{cases} \tag{60}$$
$$\Rightarrow 2 \times xsec > xsec + 1 > 2$$

Moreover, we know that the number of resources is a positive integer, which leads to $N \geqslant 1$. So, considering Equation 60, we have Equation 61.

$$(xsec + 1)^N < (2 \times xsec)^N \Rightarrow |\mathcal{S}_{DimRed}| < |\mathcal{S}_{Naive}| \tag{61}$$

□

According to Figure 5, the acceptance ratio achieved by Δ_{DimRed} is significantly higher than that of Δ_{Naive}. Moreover, its performance does not downgrade too much with the growth of the problem scale.

2) *Ranking conversion:* Similar to the dimension reduction technique, the number of observation states is reduced in the ranking conversion technique. But this time, the number of input variables are not changed. In this technique, the values of the variables with the same category are first sorted, and then, their rankings are used in constructing the states. For example, suppose that there are three physical resources in a network, and their supported security levels are as $\{3, 2, 5\}$. The ranking conversion technique changes these values into $\{2, 3, 1\}$. If in another state, the security levels are as $\{2, 0, 7\}$, this technique produces the same state. This is an example of how a single state is assigned to multiple states with nearly similar situations by this technique. So, the agent observes repeated states frequently and can enhance its scalability.

To reduce the number of observation states, SIRL [70, 71] has used the ranking conversion technique. Instead of presenting the pure numbers in the environment states, their ranking is compared to other nodes.

Now, we suggest an RL model, called Δ_{RankCo}, that uses the technique of ranking conversion to suggest an optimal mapping solution for Scenario 2. The security levels of the resources are presented by their rank in this model. The states of this model are formed based on Equation 62.

Model 22: Δ_{RankCo}

$$\mathcal{S} = \begin{bmatrix} secra_1 & secra_2 & \dots & secra_N \\ vd_1 & vd_2 & \dots & vd_N \end{bmatrix} \tag{62}$$

N	:	*The total number of physical resources*
$secra_i$	:	*The ranking of the supported security level of the i^{th} physical node in the set of all physical nodes*
vd_i	:	*A binary value, which is one if the current virtual node can be mapped on the i^{th} physical node*

Again, we must show that the number of possible states in Δ_{RankCo} is less than that of Δ_{Naive}. It must be noted that for the cases, where $xsec$ is lower than N, this model does not reduce the states. However, as the values of $xsec$ are 10 and 500 in Scenario 1 and Scenario 2, respectively, and $N = 7$, we can assume $xsec > N$ (Theorem 2).

Theorem 2. *The number of states the agent may face in Δ_{RankCo} is less than that of Δ_{Naive} when $xsec > N$. In other words, assuming that $xsec > N$, we have $|\mathcal{S}_{RankCo}| < |\mathcal{S}_{Naive}|$.*

Proof. As discussed in the proof of Theorem 1, the number of possible states in Δ_{Naive} is $(xsec \times 2)^N$. On the other hand, each pair in the states of Δ_{RankCo} consists of a binary value and a number that varies from 1 to N, as it is a ranking value. Hence, each pair can have $(N \times 2)$ different values, and the number of states is $(N \times 2)^N$. According to the assumption of $N < xsec$, we have Equation 63.

$$\begin{aligned} xsec > N \Rightarrow & \begin{cases} 2 \times xsec > 2 \times N \\ N \geqslant 1 \end{cases} \\ & \Rightarrow (2 \times xsec)^N > (2 \times N)^N \\ & \Rightarrow |\mathcal{S}_{Naive}| > |\mathcal{S}_{RankCo}| \end{aligned} \tag{63}$$

□

Based on the evaluation results reported in Figure 5, the performance of Δ_{RankCo} is higher than that of Δ_{DimRed}.

3) Action limitation: Another way to reduce the number of states in an RL model is to restrict the actions in a way that automatically considers some of the information required for presenting the states by avoiding specific actions. This technique is action limitation, and it is a kind of dimension reduction, but with manipulation in the action space too. Based on this technique, when a specific action has to be avoided, its probability of being selected becomes zero. Hence, there is no need to consider the input variables, which are merely used to illustrate the features related to unsuccessful actions.

SRA-DRL [83] uses this technique for improving its scalability. Its agent needs the delay constraint to perform valid allocations. This constraint is first provided by the state, but for scalability issues, it is removed, and the agent is not allowed to perform the actions that lead to allocations, which do not satisfy the constraints.

We suggest Δ_{ActLim} for Scenario 2 using this technique. In this model, the resources that cannot support the requested security level or have previously mapped requests are avoided. To be more precise, the probability distribution of the actions in Δ_{ActLim} is as Equation 64.

Model 23: Δ_{ActLim}

$$\pi(a \mid s) = \begin{cases} 0, & vd_a = 0 \\ \dfrac{e^{Q(s,a)/\psi}}{\sum\limits_{a \in \mathcal{A}} e^{Q(s,a)/\psi}}, & vd_a = 1 \end{cases} \tag{64}$$

$$\mathcal{S} = [sec_1 \quad sec_2 \quad \dots \quad sec_N] \tag{65}$$

$\mathcal{A}$	:	*The action space*
s	:	*The current state*
a	:	*The selected action*
ψ	:	*The temperature parameter of the Boltzmann distribution*
sec_i	:	*The supported security level of the i^{th} physical node*
vd_a	:	*A binary value, which is one if the current virtual node can be mapped on the selected resource (i.e., a^{th} physical node)*

Now that the capability of the resources for hosting a request is considered by the action space, we can omit their validity in the state structure (Equation 65). To show that the number of states in Δ_{Naive} is greater than that of Δ_{ActLim}, we prove Theorem 3.

Theorem 3. *The number of states the agent may face in Δ_{ActLim} is less than that of Δ_{Naive}. In other words, we have $|\mathcal{S}_{ActLim}| < |\mathcal{S}_{Naive}|$.*

Proof. As discussed in the proof of Theorem 1, the number of different states in Δ_{Naive} is $(xsec \times 2)^N$. As each element in the state structure of Δ_{ActLim} varies from one to $xsec$, the number of different states is $xsec^N$. Recalling that $xsec > 1$, we have Equation 66.

$$\begin{cases} xsec + xsec > 1 + xsec \\ xsec + 1 > xsec \end{cases} \Rightarrow 2 \times xsec > xsec > 1 \tag{66}$$

Now, we can conclude Equation 67.

$$\begin{aligned} & \left.\begin{matrix} 2 \times xsec > xsec > 1 \\ N \geqslant 1 \end{matrix}\right\} \Rightarrow xsec^N < (2 \times xsec)^N \\ & \qquad \Rightarrow |\mathcal{S}_{Naive}| > |\mathcal{S}_{ActLim}| \end{aligned} \tag{67}$$

□

Figure 5 claims that the technique of action limitation can greatly increase the scalability of our model for Scenario 2. We can also see that the agent in Δ_{ActLim} learns sooner (i.e., in episode 250) than the other agents.

4) Accuracy sacrifice: The main focus of the accuracy sacrifice technique is on the trade-off between the scalability and accuracy in designing an RL model. Based on this method, the agent sacrifices some information in exchange for improved scalability.

STE-SDN [91] is one of the models using this technique. The agent in this model has to find the secure paths in a network, and to have a complete view of the network topology; it is required to present all the edges in the environment states. However, the number of edges grows as the number of nodes is increased. As a result, STE-SDN suggests reducing the number of states by only presenting the top candidate paths to the agent.

We define Δ_{AccSac} as a learning model for Scenario 2, that utilizes the accuracy sacrifice technique. In this model, we ignore the security levels of the resources and just provide their validity to the agent. The state structure of Δ_{AccSac} is shown in Equation 68.

Model 24: Δ_{AccSac}

$$\mathcal{S} = [vd_1 \quad vd_2 \quad \dots \quad vd_N] \tag{68}$$

N	:	*The total number of physical resources*
vd_i	:	*A binary value, which is one if the current virtual node can be mapped on the i^{th} physical node*

Now, we are going to prove that the number of states in Δ_{Naive} is greater than that of Δ_{AccSac} (Theorem 4).

TABLE VI
COMPARISON OF RL TECHNIQUES FOR NETWORK SECURITY

Paper	Motivation	Attack Type(s)	RL Technique(s)	Targeted Security Aspect	Scalability	Performance Metrics	Limitations	Future Research Directions
Our Paper	Survey and tutorial on RL for network security.	DoS/DDoS, Authentication, General attacks.	Hybrid RL models, Game-theory RL, Deep RL.	Network defense, Privacy, Intrusion detection.	Solutions with Federated Learning.	Detection accuracy, Attack mitigation.	Case studies and real-world deployments.	Need for further testing in dynamic, real-world environments.
	Hands-on tutorial for RL model training.	Traditional attacks.	RL with dynamic reward.	Defense strategies for networks.	Large-scale networks, but needs further optimization.	RL adaptability, Action space design.	Evaluation of RL models in high-latency, real-time environments.	-
Xiaozhen Lu et al. (2024)	Focus on 6G PHY-layer security.	Eavesdroppers, Spoofers, Inference attacks.	RL with Game-theory and Policy RL.	PHY-layer security, Privacy.	Theoretical models; not fully scalable.	High accuracy in PHY-layer jamming mitigation.	Focuses on PHY-layer, limited generalization.	Multi-layer solutions for heterogeneous networks.
	Security in UAV networks.	-	Game-theory-driven RL strategies.	Resource allocation.	Not evaluated in large-scale deployments.	RL-based attack prediction and mitigation.	Testing in larger, diverse networks.	-
Jesús F. Cevallos M. (2024)	Intrusion detection for IoT using DRL.	IoT-specific attacks, Privacy breaches.	DRL, Policy-based RL.	Intrusion detection, Privacy preservation.	Limited to IoT with low scalability.	Edge computing, Low power consumption.	Focused on IoT, Lack of network security.	Expand to other environments like smart cities.
	Focus on feature extraction in IoT.	Edge-based, Efficient security.	DRL-based feature extraction.	Real-time threat detection, Privacy issues.	Scalability issues for high-dimensional threat spaces.	Real-time attack detection with low computation.	Exploration of more scalable RL models for IoT.	-
Hamza Kheddar et al. (2024)	IDSs for Network infrastructure.	IoT network threats.	Deep RL, Q-learning, DQN.	Network security.	Limited scalability to non-ICS.	IDS detection rate, False-positive/negative rate.	Lack of detailed case studies and real-world implementation.	Extend RL-based IDS models to large-scale systems.
	Explore DRL for Intrusion Detection in critical systems.	Network intrusion, Malware, DDoS attacks.	Q-learning for IDS optimization.	IoT, ICS, and Smart Grid security.	Scalable in small systems, not in large multi-domain environments.	Real-time intrusion response times.	Scalability to complex multi-layer IDS systems.	-
This Survey's Contribution	RL-based approach for network security across diverse threats.	DoS/DDoS, Epidemic, Authentication.	Hybrid RL, Federated Learning, Game-theory-based RL.	Comprehensive security, Intrusion detection, Privacy protection.	Scalable design, handles large and diverse network topologies.	High detection accuracy, Fast response to new threats.	Need for real-world validation, Practical case studies.	Further optimization of RL models for real-world deployment.
	Hands-on RL model design.	Explores emerging attacks and new network architectures.	Deep RL-based multi-layer defense.	Scalable solutions for future networks.	-	Adaptability to dynamic, high-latency networks.	-	-

Theorem 4. *The number of states the agent may face in Δ_{Naive} is greater than that of Δ_{AccSac}. In other words, we have $|\mathcal{S}_{Naive}| > |\mathcal{S}_{AccSac}|$.*

Proof. We know from the proof of Theorem 1 that the number of states in Δ_{Naive} is $(xsec \times 2)^N$. On the other hand, the elements constructing the states of Δ_{AccSac} are binary values, and hence, they have only two possible values. As a result, the number of states in this model is 2^N. Recalling that $1 < xsec$, we conclude that $2 < 2 \times xsec$. As N is greater than or equal to one, we have $2^N < (2 \times xsec)^N$. Accordingly, we have $|\mathcal{S}_{AccSac}| < |\mathcal{S}_{Naive}|$. □

The evaluation results show that the acceptance ratio achieved by utilizing Δ_{AccSac} is nearly the same as the acceptance ratio of Δ_{ActLim}, and they are both the top techniques for Scenario 2. This high performance does not always appear when using the accuracy sacrifice technique. However, in our scenario, due to the high learning performance of Δ_{AccSac}, which is illustrated in Figure 4, a significantly high performance is achieved by this technique. The agent in Δ_{AccSac} learns later than the agent in Δ_{ActLim}, as the lines indicating the average undiscounted reward in Figure 4 reach their maximum at episodes 250 and 1000 for Δ_{ActLim} and Δ_{AccSac}, respectively.

D. Security

As we focus on the potential of RL models to secure digital networks, the most important aspect to keep in mind is the security of the model itself. Technology goes forward daily, and it is impossible to avoid its advancement in helping malicious users and adversaries who aim to target critical assets in the network. As security approaches are improved using the new technologies, the adversarial tools are also improved using the same technology. This is why securing the security tools is of great importance.

There are different types of threats against RL models [121]. Some of these attacks are launched to disturb the training phase of these models. In a poisoning attack, the adversary manipulates the reward function so that the agent learns something else rather than the network defender's goal. In a worse case, the adversary forces the agent to learn malicious behavior. One of the methods for protecting the learning model against reward poisoning attacks is to define the reward function in a way that adding values lower than a threshold cannot result in misleading the agent [122].

There are also some other threats against RL models, in which another RL model is trained to find the parameters for destructive impacts on the target model. An example of this type of attack is a critical point attack, in which the adversary determines the most effective method to launch the attack [123].

V. Comparison of RL-based Network Security surveys

This section summarizes the key contributions of existing surveys, as well as their limitations and what they have done, and how their work differs from the present paper. In recent years, several surveys [124, 125, 126] have been reviewed in network security (TABLE III). We briefly summarize their contributions and identify the limitations that our survey aims to address. In [125] Security and Privacy, this survey examines the application of RL to enhance physical (PHY) and interlayer security in 6G networks. It focuses on threats such as eavesdropping and spoofing, and examines the effectiveness of RL-based solutions in addressing these challenges. Although comprehensive in the context of 6G, this work is limited to PHY layer threats and does not address the broader network security challenges beyond 6G. Also, RL for Intrusion Detection in the Internet of Things has been presented in [126]. this review approaches designed for the unique needs of IoT networks, such as low power consumption, privacy preservation, and optimization. Although it provides an in-depth classification of design alternatives for IoT IDS, it is limited to specific IoT contexts. In [124] presents artificial intelligence techniques in IDS for industrial control systems. it focuses on specific network environments and lacks a comprehensive tutorial or practical guidance for implementing RL-based security models. It categorizes the existing research into DRL-only, IDS-only, and DRL-based IDS approaches and provides a comparative analysis.

A. Comparative Analysis of RL Models for Network Security

This section presents a comparative analysis of the leading RL-based models, highlighting their respective strengths, limitations, and practical implications. The insights gained from this analysis support the selection of optimal models tailored to diverse network security.

- **Adaptability and Flexibility:** Deep RL and actor-critic based models (e.g., SIRL [70, 71]) exhibit high adaptability to dynamic network conditions, learning robust defense strategies in evolving threat landscapes. On the other hand, models like DRL-IPS [72] and Werewolf-RL [73] are tailored for specific attack types or network structures, offering targeted solutions but potentially limited generalizability.
- **Scalability:** Architectures such as DRWD [79] and SRA-DRL [83] are designed for large-scale network environments, using dimension reduction and efficient state encoding to mitigate the curse of dimensionality. Models relying on tabular Q-learning (e.g., ADRLO [84]) may struggle as the network size increases.
- **Detection Accuracy:** Hybrid and deep learning-enhanced RL models (e.g., DRL-IMF [82], DRL-SDS [87], SparseIDS [88]) achieve high detection rates for both known and zero-day threats due to their feature extraction capability. Value-based and policy-based models without deep components may exhibit lower detection accuracy for complex attack patterns.
- **Robustness Against Attack Sophistication:** Adversarial RL, multi-agent, and hybrid models (e.g., SIRL, DREVAN [78]) are more robust against adaptive adversaries and multi-stage attacks. They can dynamically adjust to new attack vectors, whereas simpler value-based models may be vulnerable to sophisticated evasion tactics.
- **Computational Efficiency:** While deep RL models provide higher performance, they require substantial computational resources, both for training and inference. Models with lighter architectures (e.g., DRWD, ADRLO) offer faster deployment and are preferable for resource-constrained environments such as IoT networks.
- **Deployment Readiness:** Approaches validated on real-world datasets (e.g., Werewolf-RL, SparseIDS, DRL-IPS) demonstrate better readiness for practical deployment. Some models require extensive simulation or synthetic data, which may not reflect real traffic and attack diversity.

Table VII further summarizes the comparative attributes of selected RL models.

B. Positioning within the Recent Literature

Several recent surveys have examined the application of reinforcement learning (RL) in network security from diverse perspectives. Many of these works have focused on specific technical directions, such as multi-agent RL, applications in software-defined networking, or deep RL for IoT security. While these reviews have provided critical overviews and established useful taxonomies, they often leave certain needs unmet, particularly in terms of practical implementation, hands-on guidance, and comprehensive coverage of emerging trends.

As detailed in Table VIII, a recurring limitation across previous surveys is their limited discussion of implementation details, the absence of step-by-step tutorials, and relatively little attention to real-world deployment scenarios. For example, structured classifications of RL algorithms and their security applications are well-represented; yet, most reviews stop short of providing actionable examples, best practices, or references to datasets and public environments that would support practitioners aiming to apply RL in cybersecurity.

The present survey aims to address these gaps. In addition to reviewing foundational concepts and the latest research advances, we have placed particular emphasis on integrating a practical tutorial, complete with illustrative examples and references to relevant datasets and simulators. This hands-on approach is intended to support both newcomers and experienced researchers in bridging the gap between theory and implementation. Moreover, our discussion includes an explicit comparison with recent literature, highlighting both the complementary aspects and unique contributions of this work, such as its operational focus and attention to challenges surrounding real-world adoption.

C. Comprehensive Comparison with Conventional Security Methods

To provide a holistic and up-to-date perspective on the effectiveness of reinforcement learning (RL) compared to traditional security solutions, we compiled recent empirical results from twenty studies conducted between 2021 and 2024. Table IX summarizes a wide spectrum of network security scenarios—including DDoS mitigation, intrusion detection, botnet identification, malware traffic classification, anomaly detection in IIoT and cloud environments, access control, zero-day attack detection, and next-generation 6G network defenses.

Each study reports comparative metrics for both conventional (e.g., heuristic, rule-based, or classical ML) and RL-based methods, covering standard benchmarks such as accuracy, precision, recall, F1-score, false positive rate (FPR), and response latency. Across these scenarios, RL-based models consistently outperform traditional counterparts, often achieving notably higher accuracy and recall with reduced false positives and competitive latency. This demonstrates the versatility and practical superiority of RL frameworks for handling complex, dynamic, and evolving security threats. For example, in DDoS mitigation on the CICDDoS2021 dataset, RL-based models improved the detection rate by over 5% compared to threshold-based approaches, while maintaining a lower FPR. Similarly, for intrusion detection tasks on widely used datasets (CSE-CIC-IDS2018, ISCX2021), RL solutions outperformed both SVM and Random Forest classifiers in all major evaluation metrics. Notably, in more advanced applications—such as anomaly detection for IIoT, 6G, and cloud computing—RL approaches demonstrated greater adaptability and robustness, with performance improvements reported in both standard and zero-day scenarios.

This comprehensive benchmarking not only validates the significant advances achieved by RL in network security, but also highlights the broad applicability and scalability of RL algorithms for present and cyber defense challenges. We believe this quantitative analysis, grounded in recent literature, offers valuable guidance for both researchers and practitioners seeking to assess the tangible benefits of RL-based methods over conventional solutions.

VI. Conclusion

With RL models as a standout example, machine learning techniques hold great promise in addressing the intricate challenges associated with fortifying computer networks against advanced and ever-evolving threats. This paper serves as a comprehensive tutorial-based survey, delving deeply into this field and offering a comprehensive exploration. In the introduction, the paper underscores the growing significance of machine learning, particularly RL models, in effectively addressing the complex issues related to network security. It highlights the critical role that RL plays in adapting to emerging threats and provides an overview of the paper's extensive coverage. The background section of the survey provides readers with a solid foundational understanding of RL. It explores key terminologies and mathematical equations relevant to various aspects of the training phase. By elucidating these fundamentals, the paper ensures that readers have the necessary grounding to engage with the subsequent material. The step-by-step tutorial is a centerpiece of the paper, guiding readers through the intricate process of constructing and training RL models tailored specifically for security applications. It imparts practical insights, enabling readers to develop and deploy RL models to safeguard network integrity. The categorization of research is a significant feature, categorizing the reviewed studies based on specific types of cyber threats that network defenses are designed to mitigate. These categories encompass diverse threats, including, but not limited to, DoS/DDoS attacks, epidemic threats, authentication vulnerabilities, and general cyber assaults. This categorization highlights the versatility of RL models in countering cyber threats. In conclusion, the survey reiterates the pivotal role of RL models in enhancing network security and their adaptability to diverse threat landscapes.

Acknowledgment

This research is partially supported by the European Union's Horizon Europe research and innovation program under the RIGOUROUS project (Grant No. 101095933).

TABLE VII
COMPARATIVE OVERVIEW OF RL MODELS FOR NETWORK SECURITY

Model	**Main Attack Focus**	**Scalability**	**Adaptability**	**Detection Accuracy**	**Real-World Readiness**
GCN-RLSec [127]	Cloud Attack Graphs	High	Moderate	High	Partial
FedRL-IDS [29]	IIoT Intrusion Detection	High	High	High	High
DQN-EdgeSec [22]	Edge DDoS, SDN	High	Moderate	High	Partial
MA-DRL-SDN [32]	DDoS, Routing	High	High	High	Partial
DQN-IDS [128]	IoT Intrusion	Moderate	High	High	High
FedDRL-Detect [129]	Distributed IDS	High	Moderate	High	Partial
Meta-DQN-IoT [35]	Adaptive IoT Attacks	Moderate	High	High	Simulation
DRL-IIoT-AD [130]	IIoT Anomaly	High	Moderate	High	Partial
GAN-DRL-5G [23]	Adversarial, 5G	Moderate	High	Moderate	Simulation
FRL-Privacy [24]	Privacy Attacks	Moderate	High	High	Partial
EdgeRL-VNF [131]	VNF Placement, Edge	High	Moderate	High	Partial
MARL-5GSec [132]	5G/6G Multi-Agent	High	High	High	Simulation
DRL-SDNDef [133]	SDN DDoS/Intrusion	High	Moderate	High	High
FedRL-CPS [134]	CPS/IIoT Attacks	High	High	High	Partial
Hybrid-RL-IDS [135]	General IDS	Moderate	High	High	High
DeepQ-DDOS [136]	DDoS/SDN	High	Moderate	High	Simulation
ActorCritic-Auth [137]	Auth. Attacks	Moderate	High	High	Simulation
RL-IoTTrust [138]	IoT Trust Mgmt.	Moderate	Moderate	High	Partial
PPO-NetSec [139]	Network-wide Attacks	High	High	High	Simulation
MADDPG-Edge [140]	Edge/IoT Collab.	High	High	High	Partial
EdgeDQN-Sec [141]	Edge Intrusion	High	Moderate	High	Partial
RL-NetShield [142]	Botnet/IoT	Moderate	High	High	Simulation
DRL-APTGuard [143]	APT Detection	High	Moderate	High	Partial
MAAC-IDS [144]	Multi-Agent IDS	High	High	High	Simulation
FedSAC-IIoT [145]	IIoT Attacks	High	High	High	Partial
D3QN-SDN [146]	SDN DDoS/Intrusion	High	Moderate	High	Simulation
TrustDRL-IoT [147]	IoT Trust	Moderate	High	High	Partial
ResRL-IDS [148]	General IDS	High	High	High	High
AdvPPO-Net [149]	Adversarial Attacks	Moderate	High	High	Simulation
HybridFedRL-Sec [150]	Distributed IDS	High	High	High	Partial
MDRL-NetDef [151]	General Attacks	High	High	High	Simulation
SARL-IDS [152]	Intrusion Detection	Moderate	High	High	Partial
GDRL-SDN [153]	SDN/IoT	High	Moderate	High	Simulation
DeepSARSA-IIoT [154]	IIoT Threats	High	High	High	Partial
MARL-CloudSec [155]	Cloud Intrusion	High	High	High	Simulation
C51-RL-IDS [156]	General IDS	Moderate	High	High	Partial
AdvRL-Edge [157]	Edge Attacks	High	High	High	Simulation
FedAC-Sec [158]	Federated Attacks	High	Moderate	High	Partial
MASAC-IoT [159]	IoT Attacks	Moderate	High	High	Partial
SafeDRL-Net [160]	Network-wide Attacks	High	High	High	Simulation

REFERENCES

[1] A. Javadpour, F. Ja'fari, T. Taleb, M. Shojafar, and C. Benzaïd, "A comprehensive survey on cyber deception techniques to improve honeypot performance," *Computers & Security*, p. 103792, 2024.

[2] A. Javadpour, F. Ja'fari, T. Taleb, Y. Zhao, Y. Bin, and C. Benzaïd, "Encryption as a service for iot: opportunities, challenges and solutions," *IEEE Internet of Things Journal*, 2023.

[3] TTCP, "Cage challenges," 2024. [Online]. Available: https://github.com/cage-challenge

[4] A. M. K. Adawadkar and N. Kulkarni, "Cyber-security and reinforcement learning—a brief survey," *Engineering Applications of Artificial Intelligence*, vol. 114, p. 105116, 2022.

[5] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[6] T. T. Nguyen and V. J. Reddi, "Deep reinforcement learning for cyber security," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[7] A. Uprety and D. B. Rawat, "Reinforcement learning for iot security: A comprehensive survey," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 8693–8706, 2020.

[8] Y. Liu, A. Halev, and X. Liu, "Policy learning with constraints in model-free reinforcement learning: A survey," in *The 30th International Joint Conference on Artificial Intelligence (IJCAI)*, 2021.

[9] Y. Du and F. Li, "Intelligent multi-microgrid energy management based on deep neural network and model-free reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 11, no. 2, pp. 1066–1076, 2019.

[10] T. M. Moerland, J. Broekens, A. Plaat, C. M. Jonker *et al.*, "Model-based reinforcement learning: A survey," *Foundations and Trends® in Machine Learning*, vol. 16, no. 1, pp. 1–118, 2023.

[11] G. Neu and C. Pike-Burke, "A unifying view of optimism in episodic reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1392–1403, 2020.

[12] K. Khetarpal, M. Riemer, I. Rish, and D. Precup, "Towards continual reinforcement learning: A review and perspectives," *Journal of Artificial Intelligence Research*, vol. 75, pp. 1401–1476, 2022.

[13] R. S. Sutton, "Reinforcement learning: An introduction," *A Bradford Book*, 2018.

TABLE VIII
DETAILED COMPARISON OF RECENT RL-BASED NETWORK SECURITY SURVEYS (2019–2025)

Survey (Year)	**Journal**	**RL Approach**	**Threats/Use Cases**	**Structured Taxonomy**	**Code/Data**	**Practical Implementation/ Tutorial**	**Key Gaps Identified**	**Main Strengths**
Finistrella et al. [161] (2025)	IEEE TNSM	Multi-Agent RL	Cybersecurity (intrusion, coordination, MARL)	Yes	No	No	Lacks practical guide, focuses on MARL	Structured MARL taxonomy, research challenges
Luong et al. [162] (2023)	IEEE CST	Deep RL, Hybrid	Network security (DDoS, intrusion, SDN, IoT)	Yes	No	No	Emerging topics underexplored	Comprehensive DL/RL coverage, future outlook
Nguyen & Reddi [163] (2022)	ACM CSUR	RL (general), DQN, Policy Gradients	Cybersecurity (algorithms, architectures)	Yes	No	No	Limited real-world validation	Algorithmic focus, taxonomy
Zhang et al. [21] (2022)	IEEE TDSC	Deep RL	IoT security, intrusion detection	Partial	No	No	SDN, blockchain not covered	IoT-specific analysis, scalability
Xu et al. [164] (2023)	IEEE CST	Deep RL	SDN/Edge, anomaly detection	Yes	No	No	Lacks tutorial, limited real testbeds	SDN-centric taxonomy, evaluation metrics
Ayoade et al. [165] (2021)	FGCS	RL (general), DQN	IoT, cloud security, access control	Yes	No	No	Lacks hands-on details	Focus on IoT/cloud, research gaps
Uprety & Rawat [166] (2020)	IEEE Access	RL (Q-learning, Sarsa, etc.)	Cybersecurity (general), SDN	Partial	No	No	Outdated domains, lacks new trends	Early review of RL for cybersecurity
Mousavi et al. [34] (2023)	Computer Networks	RL (Q-learning, DQN)	SDN security, DoS/DDoS, routing	Partial	No	No	No practical demo, limited to SDN	Recent SDN/DoS focus, attack graphs
He et al. [167] (2024)	IEEE WC	RL (DQN, Policy-based, Actor-Critic)	5G/6G/IRS, next-gen wireless threats	Yes	No	No	No tutorial, code/dataset missing	5G/6G state-of-art, future vision
Wang et al. [168] (2023)	IEEE IoT J	RL (general), DRL	Anomaly detection, IIoT, CPS	No	No	No	Lacks taxonomy, deployment	IIoT/CPS analysis, anomaly methods
Zhang et al. [128] (2022)	IEEE TDSC	Deep RL, GNN	Distributed intrusion detection in IoT	Yes	No	No	Scalability in large-scale IoT	Robust DRL-based cooperative detection
Li et al. [22] (2022)	Computer Networks	Multi-agent DRL	DDoS mitigation in SDN	Yes	No	No	Limited testbed evaluation	Distributed attack mitigation
Kim et al. [23] (2021)	JISAS	Adversarial RL, GANs	Adversarial attack defense in 5G	No	No	No	GAN robustness not fully validated	RL-based GAN for adversarial defense
Wang et al. [24] (2023)	IEEE TIFS	Federated RL	Privacy-preserving anomaly detection	Yes	No	No	Scalability in federated setting	Scalable federated RL detection
Chen et al. [127] (2024)	IEEE TIFS	Graph RL, GCN	Attack path prediction in cloud	Yes	No	No	Attack graph scalability	GCN-enhanced analytics
Xu et al. [133] (2023)	IEEE Comms Surveys	Deep RL	Security in SDN	Yes	No	No	Practical case studies missing	SDN-focused survey
Huang et al. [29] (2021)	IEEE IoT J.	Federated Deep RL	Intrusion detection in IIoT	Partial	No	No	Edge deployment challenges	Collaborative intrusion detection
Sun et al. [30] (2022)	IEEE TNSM	Adaptive RL	IoT anomaly detection	No	No	No	Lack of robust evaluation	Dynamic IoT anomaly detection
Zheng et al. [169] (2021)	IEEE TIFS	Deep RL	APT detection in CPS	No	No	No	Generalization to unseen attacks	Detecting APTs in CPS
Lin et al. [32] (2023)	IEEE TWC	Multi-agent DRL	Wireless network security	Yes	No	No	Joint resource-security challenge	Resource allocation, security enhancement
Liu et al. [170] (2022)	JISA	Distributed RL	DDoS defense in cloud	Yes	No	No	Privacy of collaborative defense	Privacy-preserving defense
Li et al. [35] (2024)	IEEE IoT J.	Hybrid DQN + Meta-learning	Adaptive IoT security	Yes	No	No	Meta-learning complexity	Adaptive security in IoT
Zhang et al. [130] (2024)	IEEE IoT J.	DRL	Anomaly detection in IIoT	Yes	No	No	IIoT data imbalance	IIoT anomaly detection
Guo et al. [129] (2023)	Computer Networks	Federated RL	Distributed IDS	Partial	No	No	Model synchronization overhead	Scalable IDS for networks
This survey	—	RL (Single/Multi-Agent, Deep RL, Hybrid)	DDoS, SDN, IoT, IIoT, Blockchain, 6G, access control	**Yes**	**Sample code/datasets referenced**	**Yes (detailed step-by-step tutorial, hands-on)**	**Bridges practical gaps, covers emerging areas, deployment, operationalization**	**Holistic, up-to-date, tutorial, practical, emerging topics, deployment-focused**

TABLE IX
COMPREHENSIVE COMPARISON OF RL-BASED AND TRADITIONAL SECURITY METHODS ACROSS RECENT STUDIES (2021–2024)

Scenario	**Dataset/Env.**	**Traditional Method**	**RL Method**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1 (%)**	**FPR (%)**	**Latency (ms)**	**Ref**
DDoS Mitigation (SDN)	CICDDoS2021	Heuristic	DQN	89.7	90.1	88.3	89.2	7.2	44	[171]
Intrusion Detection	CSE-CIC-IDS2018	Random Forest	DDPG	93.4	94.1	93.6	93.8	4.9	61	[172]
Botnet Detection	ISCX2021	SVM	Deep Q-Learning	90.2	91.5	89.3	90.4	6.0	57	[173]
Malware Traffic Detection	USTC-TFC2016	NB Tree	PPO	95.1	94.6	95.2	94.9	3.5	51	[174]
Ransomware Detection	EMBER2018	Logistic Regression	A2C	93.8	92.5	93.1	92.8	5.1	63	[175]
IIoT Anomaly Detection	WUST-IDS2022	Isolation Forest	DDPG	92.7	91.9	92.2	92.0	4.4	59	[176]
DNS Tunneling Detection	CIC-DNS2021	Decision Tree	Rainbow DQN	91.5	90.2	91.4	90.8	5.8	49	[177]
Access Control	IEEEDataPort2022	RBAC	Deep RL (SAC)	94.3	94.9	93.7	94.3	4.6	46	[178]
6G Network Attack Defense	Custom-6GSim	ML Ensemble	DRL (A3C)	95.2	94.1	95.4	94.7	3.3	38	[179]
Zero-Day Attack Detection	BoT-IoT	KNN	DRL (DDPG)	92.1	91.6	92.8	92.2	4.7	53	[180]
DDoS Mitigation (SDN)	SDN-DDoS-2022	Threshold-based	DDPG	91.2	90.4	90.8	90.6	6.1	48	[181]
IoT Intrusion Detection	TON-IoT	Signature-based	DRL (PPO)	93.5	93.0	92.8	92.9	5.3	54	[182]
Botnet Detection	CTU-13	SVM	Double DQN	92.7	92.1	92.4	92.2	4.2	45	[183]
Malware Traffic Classification	CICMal2019	Decision Tree	DRL (CNN-DQN)	94.8	95.2	94.5	94.8	4.6	52	[184]
Anomaly Detection (IIoT)	WUST-IDS2022	Random Forest	Adaptive RL	94.2	93.8	94.0	93.9	3.8	50	[185]
ICS Intrusion Detection	SWaT-2023	Naive Bayes	DRL (A2C)	93.3	93.1	92.6	92.8	5.4	59	[186]
Cloud Anomaly Detection	CloudSim-IDS	One-Class SVM	DRL (DQN)	92.9	93.5	92.3	92.9	4.9	48	[187]
Anomaly Detection (Smart Grid)	SmartGridDS	SVM	PPO	91.6	91.1	91.8	91.4	6.5	62	[188]
Access Control (5G)	5G-AuthSim	RBAC	RL (DDPG)	94.6	94.3	94.1	94.2	3.6	43	[189]
Zero-Day Detection (IoT)	BoT-IoT	NB Tree	Deep RL (PPO)	93.1	92.5	92.9	92.7	4.3	50	[190]

[14] A. Kanervisto, C. Scheller, and V. Hautamäki, "Action space shaping in deep reinforcement learning," in *2020 IEEE conference on games (CoG)*. IEEE, 2020, pp. 479–486.

[15] C. Rong, J. Paramesh, and L. R. Carley, "A deep reinforcement learning framework for high-dimensional circuit linearization," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 9, pp. 3665–3669, 2022.

[16] J. Delaney, S. Dowey, and C.-T. Cheng, "Reinforcement-learning-based robust resource management for multi-radio systems," *Sensors*, vol. 23, no. 10, p. 4821, 2023.

[17] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, "Q-learning algorithms: A comprehensive classification and applications," *IEEE access*, vol. 7, pp. 133 653–133 667, 2019.

[18] A. Zanette, M. J. Wainwright, and E. Brunskill, "Provable benefits of actor-critic methods for offline reinforcement learning," *Advances in neural information processing systems*, vol. 34, pp. 13 626–13 640, 2021.

[19] C. Dann, Y. Mansour, M. Mohri, A. Sekhari, and K. Sridharan, "Guarantees for epsilon-greedy reinforcement learning with function approximation," in *International conference on machine learning*. PMLR, 2022, pp. 4666–4689.

[20] L. Pan, Q. Cai, Q. Meng, W. Chen, L. Huang, and T.-Y. Liu, "Reinforcement learning with dynamic boltzmann softmax updates," *arXiv preprint arXiv:1903.05926*, 2019.

[21] Y. Zhang, S. Xie, and J. Wang, "Deep reinforcement learning-based cooperative detection for iot security: A graph neural network approach," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 5, pp. 3502–3514, 2022.

[22] R. Li, X. Chen, and H. Wang, "Multi-agent deep reinforcement learning for ddos mitigation in sdn," *Computer Networks*, vol. 210, p. 108944, 2022.

[23] J. Kim, K. Lee, and M. Park, "Adversarial reinforcement learning against adversarial attacks: Applications in 5g security," *Journal of Information Security and Applications*, vol. 59, p. 102809, 2021.

[24] H. Wang, X. Zhang, and Y. Liu, "Federated reinforcement learning for privacy-preserving anomaly detection in smart networks," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 4245–4257, 2023.

[25] R. Patel and B. Mehta, "Reinforcement learning-based dynamic access control in software-defined networks," *Computer Networks*, vol. 181, p. 107580, 2020.

[26] Y. Chen, J. Sun, and K. Ren, "Graph reinforcement learning with gcn for attack path prediction in cloud networks," *IEEE Transactions on Information Forensics and Security*, vol. 19, no. 3, pp. 780–794, 2024.

[27] V. Singh, A. Prakash, and S. Sharma, "Decentralized reinforcement learning for scalable iot device authentication," *Journal of Information Security and Applications*, vol. 74, p. 103576, 2023.

[28] W. Xu, H. Li, and Y. Wang, "A comprehensive survey on deep reinforcement learning-based security in software-defined networks," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1608–1640, 2023.

[29] Q. Huang, G. Chen, and Y. Li, "Federated deep rl for intrusion detection in iiot," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9716–9728, 2021.

[30] J. Sun, Y. Xu, and Y. Luo, "Adaptive rl for dynamic iot anomaly detection," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2431–2443, 2022.

[31] B. Zheng, T. Tang, and L. Wang, "Deep reinforcement learning for the detection of advanced persistent threats in cyber-physical systems," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1927–1942, 2021.

[32] C. Lin, H. Jiang, and Y. Sun, "Multi-agent deep rl for wireless network security," *IEEE Transactions on Wireless Communications*, vol. 22, no. 6, pp. 3952–3965, 2023.

[33] F. Liu, X. Yang, and Y. Chen, "Privacy-preserving collabora-

tive defense against ddos attacks using distributed rl in cloud computing," *Journal of Information Security and Applications*, vol. 68, p. 103204, 2022.

[34] A. Mousavi, T. Firoozi, and M. Yousefi, "Rl-sdn: Reinforcement learning-based secure routing and defense against dos attacks in sdn," *Computer Networks*, vol. 231, p. 109668, 2023.

[35] T. Li, J. Sun, and L. Chen, "Hybrid dqn and meta-learning for adaptive iot security," *IEEE Internet of Things Journal*, vol. 11, no. 4, pp. 3452–3463, 2024.

[36] Y. Bai, S. Wu, and Y. Wang, "Proactive defense for smart grid communications: Deep reinforcement learning-based secure topology reconfiguration," *IEEE Transactions on Smart Grid*, vol. 14, no. 4, pp. 3101–3111, 2023.

[37] J. Hu, L. Zhao, and M. Zhang, "Cooperative deep reinforcement learning for dynamic network security risk assessment," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 2, pp. 1276–1288, 2023.

[38] Y. Zhou, X. Ma, and et al., "Adaptive security policy optimization in software-defined networks via reinforcement learning," *Future Generation Computer Systems*, vol. 144, pp. 187–196, 2023.

[39] K. Bao, S. Wang, and et al., "Multiagent deep reinforcement learning for security enhancement in smart grids," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6752–6764, 2021.

[40] L. Gao, C. Xu, and et al., "Joint deep reinforcement learning and resource allocation for secure mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 19, no. 10, pp. 2378–2392, 2020.

[41] M. T. Rahman, P. C. v. Oorschot, and et al., "Detection of iot malware using a hybrid cnn and reinforcement learning approach," *Computer Networks*, vol. 209, p. 108945, 2022.

[42] X. He, X. Wang, and et al., "Deep reinforcement learning for insider threat detection in cloud systems," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2111–2126, 2021.

[43] X. Wu, Y. Liu, and et al., "Energy-aware reinforcement learning for secure routing in wireless sensor networks," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 2, pp. 1170–1181, 2023.

[44] X. Yu, J. Li, and et al., "Meta-reinforcement learning for cyber-attack defense in v2x networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 4, pp. 4047–4057, 2023.

[45] Y. Wu, S. Li, and Q. Wang, "Rl-enabled blockchain for secure collaborative data sharing in iiot," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 1759–1771, 2023.

[46] L. Zhao, J. Chen, and Y. Xu, "Deep reinforcement learning-based adaptive intrusion detection for connected vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1600–1610, 2022.

[47] S. S. Hassan, M. Rehmani, and J. Chen, "Reinforcement learning for security and management in 6g: Challenges and opportunities," *IEEE Network*, vol. 35, no. 4, pp. 300–307, 2021.

[48] M. Tang, L. Zhang, and J. Liu, "Transfer reinforcement learning for cross-domain iot security enhancement," *Computer Networks*, vol. 209, p. 108945, 2022.

[49] H. Sadeghi and M. Mozaffari, "Real-time security adaptation using online reinforcement learning," *IEEE Access*, vol. 10, pp. 49 257–49 270, 2022.

[50] A. Gupta and P. Raj, "Phishing attack detection using deep reinforcement learning: A survey and new perspectives," *Journal of Information Security and Applications*, vol. 76, p. 103746, 2023.

[51] S. Park and Y. Kim, "Hybrid reinforcement learning and lstm model for dynamic botnet detection," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4506–4517, 2021.

[52] L. Zhang, M. Wei, and B. Sun, "Rl-based adaptive access and threat management for industrial iot security," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 7, pp. 4831–4842, 2022.

[53] J. Kim and S. Choi, "Adversarial reinforcement learning for robust network defense," *Pattern Recognition*, vol. 142, p. 110558, 2023.

[54] C. He, Y. Wang, and P. Zhang, "Meta-reinforcement learning for rapid adaptation to unknown cyber threats," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 2, pp. 1233–1245, 2024.

[55] I. Ali, S. Ahmed, and M. Imran, "Collaborative reinforcement learning for privacy-preserving iot security," *IEEE Internet of Things Journal*, vol. 9, no. 14, pp. 12 188–12 199, 2022.

[56] Y. Luo, Q. Li, and J. Zhang, "Federated reinforcement learning for anomaly detection in edge-cloud networks," *IEEE Transactions on Mobile Computing*, vol. 22, no. 1, pp. 132–144, 2023.

[57] X. Shen, L. Wang, and J. Yang, "Adaptive reinforcement learning for attack graph-based cybersecurity analysis," *Computer Networks*, vol. 195, p. 108187, 2021.

[58] C. Meng, F. Liu, and L. Jiang, "A hybrid rl approach for malware detection in software-defined networks," *Future Generation Computer Systems*, vol. 152, pp. 675–684, 2024.

[59] Q. Wang, W. Wu, and L. Xie, "Hierarchical reinforcement learning for multi-layer cyber-physical security management," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3459–3471, 2021.

[60] S. Garg, D. Madaan, R. Chauhan *et al.*, "Explainable artificial intelligence for network anomaly detection: A deep learning approach," *IEEE Access*, vol. 11, pp. 67 218–67 229, 2023.

[61] B. Liang, J. Wang, and C. Yan, "Multi-objective deep reinforcement learning for threat mitigation in sdn," *Future Generation Computer Systems*, vol. 134, pp. 56–67, 2022.

[62] L. Ma, J. Zhao, and Y. Wang, "Federated reinforcement learning for distributed intrusion detection in iot," *Journal of Information Security and Applications*, vol. 61, p. 102927, 2021.

[63] X. Dong, H. Liu, and F. Yu, "Secure communications in iot: Reinforcement learning with homomorphic encryption," *Computer Networks*, vol. 177, p. 107313, 2020.

[64] W. Jin, R. Li, and X. Liu, "Deep reinforcement learning for secure drone communication in vanets," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 17 341–17 352, 2022.

[65] Z. Deng, L. Zhang, and Y. Lin, "Adversarial reinforcement learning for resilient anomaly detection in iiot," *IEEE Internet of Things Journal*, vol. 10, no. 3, pp. 2367–2377, 2023.

[66] X. Fan, Y. Li, and B. Wang, "Deep reinforcement learning for adaptive intrusion prevention in wireless networks," *IEEE Access*, vol. 8, pp. 204 552–204 564, 2020.

[67] X. Cheng, L. Wu, and D. Song, "Reinforcement learning for cyber insurance optimization in network security investments," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3562–3574, 2021.

[68] J. Han, X. Wu, and Y. Ma, "Double dqn-based real-time traffic anomaly detection in sdn," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2762–2773, 2022.

[69] S. Raza, S. Akhtar, and Z. Abbas, "Ai-driven reinforcement learning for proactive ddos defense in edge computing," *Computer Communications*, vol. 214, pp. 91–102, 2024.

[70] A. Javadpour, F. Ja'fari, T. Taleb, and C. Benzaïd, "Enhancing 5g network slicing: Slice isolation via actor-critic reinforcement learning with optimal graph features," in *GLOBECOM 2023-2023 IEEE Global Communications Conference*. IEEE, 2023, pp. 31–37.

[71] A. Javadpour, F. Ja'fari, T. Taleb, and C. Benzaïd, "Reinforcement learning-based slice isolation against ddos attacks in beyond 5g networks," *IEEE Transactions on Network and*

Service Management, 2023.

[72] N. M. Yungaicela-Naula, C. Vargas-Rosales, J. A. Pérez-Díaz, and D. F. Carrera, “A flexible sdn-based framework for slow-rate ddos attack mitigation by using deep reinforcement learning,” *Journal of Network and Computer Applications*, vol. 205, p. 103444, 2022.

[73] A. Javadpour, F. Ja’fari, T. Taleb, H. Ahmadi, and C. Benzaïd, “Cybersecurity fusion: Leveraging mafia game tactics and reinforcement learning for botnet detection,” in *GLOBECOM 2023-2023 IEEE Global Communications Conference*. IEEE, 2023, pp. 6005–6011.

[74] A. Javadpour, F. Ja’fari, T. Taleb, and C. Benzaïd, “Detecting malicious nodes using game theory and reinforcement learning in software-defined networks,” *International Journal of Information Security*, vol. 24, no. 3, pp. 1–45, 2025.

[75] A. Javadpour, P. Pinto, F. Ja’fari, and W. Zhang, “Dmaidps: A distributed multi-agent intrusion detection and prevention system for cloud iot environments,” *Cluster Computing*, vol. 26, no. 1, pp. 367–384, 2023.

[76] A. Javadpour, F. Ja’fari, T. Taleb, and C. Benzaïd, “Reinforcement learning-based slice isolation against ddos attacks in beyond 5g networks,” *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, pp. 3930–3946, 2023.

[77] A. Javadpour, F. Ja’fari, T. Taleb, and C. Benzaïd, “5g slice mutation to overcome distributed denial of service attacks using reinforcement learning,” in *2024 17th International Conference on Security of Information and Networks (SIN)*. IEEE, 2024, pp. 1–9.

[78] Q. Zhang, J.-H. Cho, T. J. Moore, and F. F. Nelson, “Drevan: deep reinforcement learning-based vulnerability-aware network adaptations for resilient networks,” in *2021 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2021, pp. 137–145.

[79] P. Li, S. Yu, and L. Xiao, “Smart grid worm detection based on deep reinforcement learning,” in *2022 IEEE/CIC International Conference on Communications in China (ICCC)*. IEEE, 2022, pp. 684–689.

[80] S. Wang, Q. Pei, J. Wang, G. Tang, Y. Zhang, and X. Liu, “An intelligent deployment policy for deception resources based on reinforcement learning,” *IEEE Access*, vol. 8, pp. 35 792–35 804, 2020.

[81] X. Lu, L. Xiao, T. Xu, Y. Zhao, Y. Tang, and W. Zhuang, “Reinforcement learning based phy authentication for vanets,” *IEEE transactions on vehicular technology*, vol. 69, no. 3, pp. 3068–3079, 2020.

[82] B. Liu, G. Xu, G. Xu, C. Wang, and P. Zuo, “Deep reinforcement learning-based intelligent security forwarding strategy for vanet,” *Sensors*, vol. 23, no. 3, p. 1204, 2023.

[83] H. Zhang, J. Wang, H. Zhang, and C. Bu, “Security computing resource allocation based on deep reinforcement learning in serverless multi-cloud edge computing,” *Future Generation Computer Systems*, vol. 151, pp. 152–161, 2024.

[84] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, “Secure computation offloading in blockchain based iot networks with deep reinforcement learning,” *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 4, pp. 3192–3208, 2021.

[85] C. Jiang, P. Zhang, C. Jiang, and P. Zhang, “Vne solution for network differentiated qos and security requirements from the perspective of deep reinforcement learning,” *QoS-Aware Virtual Network Embedding*, pp. 61–84, 2021.

[86] ——, “Vne solution for network differentiated qos and security requirements from the perspective of deep reinforcement learning,” *QoS-Aware Virtual Network Embedding*, pp. 61–84, 2021.

[87] Y. Li and Y. Qin, “Real-time cost optimization approach based on deep reinforcement learning in software-defined security middle platform,” *Information*, vol. 14, no. 4, p. 209, 2023.

[88] M. Bachl, F. Meghdouri, J. Fabini, and T. Zseby, “Sparseids: Learning packet sampling with reinforcement learning,” in *2020 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2020, pp. 1–9.

[89] H. Moudoud and S. Cherkaoui, “Empowering security and trust in 5g and beyond: A deep reinforcement learning approach,” *IEEE Open Journal of the Communications Society*, 2023.

[90] H. Li, Y. Guo, P. Sun, Y. Wang, and S. Huo, “An optimal defensive deception framework for the container-based cloud with deep reinforcement learning,” *IET Information Security*, vol. 16, no. 3, pp. 178–192, 2022.

[91] A. Mudgerikar and E. Bertino, “Intelligent security aware routing: Using model-free reinforcement learning,” in *2023 32nd International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2023, pp. 1–10.

[92] “Stable baselines3,” https://stable-baselines3.readthedocs.io/, 2023.

[93] E. Liang, R. Liaw, R. Nishihara *et al.*, “Rllib: Scalable reinforcement learning,” *arXiv preprint arXiv:1712.09381*, 2017.

[94] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” *arXiv preprint arXiv:1606.01540*, 2016.

[95] “Mininet: An instant virtual network on your laptop,” http://mininet.org/, 2023.

[96] “Ns-3: Discrete-event network simulator,” https://www.nsnam.org/, 2023.

[97] “Omnet++ discrete event simulator,” https://omnetpp.org/, 2023.

[98] “Cicids2017 intrusion detection evaluation dataset,” https://www.unb.ca/cic/datasets/malmem-2022.html, 2017.

[99] “Nsl-kdd dataset for network intrusion detection,” https://www.unb.ca/cic/datasets/nsl.html, 2015.

[100] “Unsw-nb15 network intrusion dataset,” https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/, 2015.

[101] Y. Zhang *et al.*, “Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms,” https://github.com/vwxyzjn/cleanrl, 2021.

[102] M. Plappert, “keras-rl: Deep reinforcement learning for keras,” https://github.com/keras-rl/keras-rl, 2016.

[103] A. Kuhnle, Y. Kilcher *et al.*, “Tensorforce: A tensorflow library for applied reinforcement learning,” https://github.com/tensorforce/tensorforce, 2017.

[104] DeepMind, “Acme: A library of reinforcement learning components and agents,” https://github.com/deepmind/acme, 2020.

[105] P. S. Castro, S. Moitra, C. Gelada, S. Kumar, and M. G. Bellemare, “Dopamine: A research framework for deep reinforcement learning,” https://github.com/google/dopamine, 2018.

[106] OpenAI, “Spinning up in deep rl,” https://spinningup.openai.com/en/latest/, 2018.

[107] J. K. Terry, M. Rowland, G. Farquhar, T. Rocktäschel, and J. Foerster, “Pettingzoo: Gym for multi-agent reinforcement learning environments,” *NeurIPS Deep RL Workshop*, 2020, https://www.pettingzoo.ml/.

[108] G. T. Inc., “Gns3 - graphical network simulator,” https://www.gns3.com/, 2023.

[109] J. Ahrenholz, C. Danilov, J. R. Davis, J. J. Elliott, and J. Strayer, “Core: A real-time network emulator,” *IEEE MILCOM*, 2008, https://www.nrl.navy.mil/itd/ncs/products/core.

[110] M. Research, “Cyberbattlesim,” https://github.com/microsoft/CyberBattleSim, 2021.

[111] U. C. Cyber, “Ton_iot datasets,” https://research.unsw.edu.au/projects/toniot-datasets, 2020.

[112] ——, “Bot-iot dataset,” https://research.unsw.edu.au/projects/bot-iot-dataset, 2018.

[113] M. L. Laboratory, “Darpa intrusion detection evaluation dataset,” https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset, 1999.

[114] U. K. Archive, “Kdd cup 1999 data,” http://kdd.ics.uci.edu/

databases/kddcup99/kddcup99.html, 1999.
[115] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Iscxids2012: Intrusion detection evaluation dataset," https://www.unb.ca/cic/datasets/ids.html, 2012.
[116] A. Ray, J. Achiam, and D. Amodei, "Safety gym," https://github.com/openai/safety-gym, 2019.
[117] OpenAI, "Multi-agent particle environments," https://github.com/openai/multiagent-particle-envs, 2018.
[118] W. Foundation, "Wireshark: Network protocol analyzer," https://www.wireshark.org/, 2023.
[119] O. N. Foundation, "Sdn openflow lab," https://www.opennetworking.org/sdn-openflow-lab/, 2022.
[120] S. James, Z. Ma, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3019–3026, 2020, https://github.com/stepjam/RLBench.
[121] C. Benzaïd and T. Taleb, "AI for Beyond 5G Networks: A Cyber-Security Defense or Offense Enabler?" *IEEE Network Magazine*, vol. 34, no. 6, Nov./Dec. 2020.
[122] X. Zhang, Y. Ma, A. Singla, and X. Zhu, "Adaptive reward-poisoning attacks against reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 225–11 234.
[123] J. Sun, T. Zhang, X. Xie, L. Ma, Y. Zheng, K. Chen, and Y. Liu, "Stealthy and efficient adversarial attacks against deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5883–5891.
[124] X. Lu, L. Xiao, P. Li, X. Ji, C. Xu, S. Yu, and W. Zhuang, "Reinforcement learning-based physical cross-layer security and privacy in 6g," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 425–466, 2022.
[125] H. Kheddar, D. W. Dawoud, A. I. Awad, Y. Himeur, and M. K. Khan, "Reinforcement-learning-based intrusion detection in communication networks: A review," *IEEE Communications Surveys & Tutorials*, 2024.
[126] J. F. Cevallos M., A. Rizzardi, S. Sicari, and A. Coen Porisini, "Deep reinforcement learning for intrusion detection in internet of things: Best practices, lessons learnt, and open challenges," *Computer Networks*, vol. 236, p. 110016, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128623004619
[127] L. Chen, Q. Wu, and Y. Zhou, "Gcn-enhanced rl for attack path prediction in cloud networks," *IEEE Transactions on Information Forensics and Security*, vol. 19, no. 5, pp. 1052–1065, 2024.
[128] L. Zhang, Y. Wang, and B. Lin, "Deep reinforcement learning for distributed intrusion detection in iot," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 4, pp. 1889–1901, 2022.
[129] P. Guo, R. Zhang, and F. Chen, "Federated rl for distributed ids in networks," *Computer Networks*, vol. 230, p. 110567, 2023.
[130] M. Zhang, H. Li, and X. Xu, "Deep rl for anomaly detection in industrial iot," *IEEE Internet of Things Journal*, vol. 11, no. 5, pp. 4980–4992, 2024.
[131] L. Ma, K. Zhou, and F. Li, "Edgerl-vnf: Rl-based vnf placement for secure edge computing," *IEEE Transactions on Network and Service Management*, vol. 21, no. 2, pp. 788–802, 2024.
[132] Y. Liu, D. Wang, and X. Zhang, "Multi-agent reinforcement learning for distributed 5g/6g security," *IEEE Wireless Communications*, vol. 30, no. 4, pp. 44–52, 2023.
[133] L. Xu, Q. Zhang, and Y. Tang, "Deep rl for security in software-defined networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1023–1049, 2023.
[134] W. Zhang, F. Lin, and S. Wu, "Fedrl-cps: Federated rl for attack mitigation in cyber-physical systems," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 7, pp. 9180–9191, 2023.
[135] M. Shao, Y. Zhang, and T. Liu, "Hybrid rl-ids: A hybrid reinforcement learning approach for intrusion detection," *Computer Networks*, vol. 230, p. 109674, 2023.
[136] R. Patil, A. Sahu, and V. Singh, "Deep q-learning based ddos detection for sdn environments," in *Proc. IEEE GLOBECOM*, 2022, pp. 1–6.
[137] J. Cai, J. He, and S. Luo, "Actor-critic rl for adaptive authentication in iot," *Future Generation Computer Systems*, vol. 144, pp. 317–328, 2024.
[138] L. Bai, K. Li, and Y. Wang, "Rl-iottrust: Trust management in iot via rl," *IEEE Internet of Things Journal*, vol. 10, no. 11, pp. 9387–9399, 2023.
[139] A. Rao and S. Kumar, "Ppo-netsec: Proximal policy optimization for network-wide security," in *Proc. IEEE ICC*, 2024, pp. 2052–2058.
[140] Q. Hu, J. Liu, and S. Wang, "Multi-agent ddpg for collaborative edge/iot security," *IEEE Internet of Things Journal*, vol. 11, no. 6, pp. 5114–5125, 2024.
[141] M. Wang, L. Zhao, and J. Lin, "Edgedqn-sec: Edge-centric dqn for intrusion detection," *Computer Networks*, vol. 239, p. 111965, 2024.
[142] H. Liu, L. Zhang, and P. Qian, "Rl-netshield: Botnet detection for iot using rl," *IEEE Access*, vol. 12, pp. 62 145–62 155, 2024.
[143] Q. Zhang, H. Zhang, and L. Chen, "Drl-aptguard: Deep rl for apt detection in network security," *IEEE Transactions on Information Forensics and Security*, vol. 19, no. 8, pp. 2097–2108, 2024.
[144] P. Sun, F. Yang, and X. Zhou, "Maac-ids: Multi-agent actor-critic for intrusion detection," *Future Generation Computer Systems*, vol. 147, pp. 365–377, 2024.
[145] Q. Li, Z. He, and Y. Xu, "Fedsac-iiot: Federated soft actor-critic for iiot security," *IEEE Internet of Things Journal*, vol. 11, no. 5, pp. 4921–4932, 2024.
[146] X. Xu, Y. Wang, and H. Tan, "D3qn-sdn: Deep double dueling q-networks for sdn security," *Computer Networks*, vol. 240, p. 112015, 2024.
[147] R. Liu, Z. Yu, and T. Hu, "Trustdrl-iot: Trust management for iot via rl," *Journal of Information Security and Applications*, vol. 75, p. 103625, 2024.
[148] L. Zhang, Z. Zhang, and F. Zheng, "Resrl-ids: Residual rl-based intrusion detection," *IEEE Access*, vol. 11, pp. 99 876–99 888, 2023.
[149] J. Bai and D. Yang, "Advppo-net: Adversarial attack mitigation via ppo," in *Proc. IEEE CNS*, 2024, pp. 174–181.
[150] S. Hu, G. Wang, and Y. Lin, "Hybridfedrl-sec: Hybrid federated rl for distributed ids," *Computer Networks*, vol. 243, p. 112145, 2024.
[151] L. Zhou, P. Yu, and H. Ma, "Mdrl-netdef: Multi-dimensional rl for network defense," *IEEE Transactions on Network and Service Management*, vol. 21, no. 3, pp. 1345–1358, 2024.
[152] J. Xu, S. Zhang, and K. Li, "Sarl-ids: Soft actor-critic rl for intrusion detection," *IEEE Access*, vol. 12, pp. 84 501–84 512, 2024.
[153] Q. Zheng, Y. Liang, and M. Pan, "Gdrl-sdn: Graph rl for sdn and iot security," *Computer Networks*, vol. 228, p. 109882, 2023.
[154] Y. Li, H. Sun, and F. Guo, "Deepsarsa-iiot: Deep sarsa rl for iiot threat detection," *Future Generation Computer Systems*, vol. 146, pp. 218–230, 2023.
[155] C. Yan, P. Zhao, and Q. Lin, "Marl-cloudsec: Multi-agent rl for cloud intrusion detection," *Journal of Cloud Computing*, vol. 13, no. 1, pp. 155–168, 2024.
[156] W. Chen, J. Wu, and X. Chen, "C51-rl-ids: Categorical rl for intrusion detection," *IEEE Access*, vol. 11, pp. 74 529–74 540, 2023.
[157] T. Wang, Y. Li, and S. Cheng, "Advrl-edge: Rl against edge security attacks," *IEEE Internet of Things Journal*, vol. 10, no. 12, pp. 9877–9888, 2023.

[158] X. Liu, F. Wang, and X. Yan, "Fedac-sec: Federated actor-critic for network security," *Computer Networks*, vol. 227, p. 109729, 2023.

[159] Y. Zhou, Z. Pan, and J. Luo, "Masac-iot: Multi-agent sac for iot attack mitigation," *Future Generation Computer Systems*, vol. 150, pp. 204–215, 2024.

[160] R. Bai, X. Huang, and S. Liu, "Safedrl-net: Safe deep rl for network-wide security," *IEEE Transactions on Network Science and Engineering*, vol. 11, no. 2, pp. 1201–1213, 2024.

[161] D. Finistrella, A. Russo, and M. Conti, "Multi-agent reinforcement learning for cybersecurity: A structured survey," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 312–335, 2025.

[162] N. C. Luong, D. T. Hoang, S. Gong, Q. V. Pham, T. A. Le, Z. Han, D. Niyato, E. Dutkiewicz, and P. Wang, "Applications of deep reinforcement learning in network security: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 1–34, 2023.

[163] N. Nguyen and V. Reddi, "Reinforcement learning for cybersecurity: Algorithms, challenges, and future directions," *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–36, 2022.

[164] X. Xu, X. Sun, S. Yang, and Y. Zhou, "Deep reinforcement learning for secure software-defined networking: A survey," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 189–215, 2023.

[165] G. Ayoade, A. Banerjee, M. Rahman, and A. Shirazi, "Reinforcement learning for iot and cloud security: A survey," *Future Generation Computer Systems*, vol. 123, pp. 168–186, 2021.

[166] S. Uprety and D. B. Rawat, "Reinforcement learning for cybersecurity: A review," *IEEE Access*, vol. 8, pp. 100 334–100 351, 2020.

[167] Z. He, J. Chen, and F. R. Yu, "Reinforcement learning for secure next-generation wireless networks: 5g and beyond," *IEEE Wireless Communications*, vol. 31, no. 1, pp. 62–69, 2024.

[168] Q. Wang, Y. Li, M. Xu, and S. Liu, "Reinforcement learning-based anomaly detection in cyber-physical and industrial iot systems: A survey," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 3100–3116, 2023.

[169] L. Zheng, Y. Li, and H. Wang, "Deep rl for apt detection in cyber-physical systems," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4162–4174, 2021.

[170] F. Liu, X. Zhang, and H. Yu, "Distributed rl for collaborative ddos defense in cloud," *Journal of Information Security and Applications*, vol. 67, p. 103217, 2022.

[171] X. Li, Y. Wang, and L. Zhou, "Ddos attack detection based on dqn in cicddos2021 dataset," *IEEE Access*, vol. 10, pp. 77 235–77 247, 2022.

[172] Q. Zhang, W. Zhao, and H. Jiang, "Adaptive deep reinforcement learning for intrusion detection using cse-cic-ids2018 dataset," *Computers & Security*, vol. 126, p. 103049, 2023.

[173] M. Ahmed, J. Hossain, and M. K. Hasan, "Botnet detection using deep q-learning: A case study with iscx2021," *Journal of Network and Computer Applications*, vol. 213, p. 103547, 2023.

[174] B. Liu, Q. Sun, and F. Luo, "Ppo-based reinforcement learning for malware traffic detection in ustc-tfc2016 dataset," *Information Sciences*, vol. 600, pp. 212–226, 2022.

[175] D. Xu, J. Li, and S. Yu, "A2c-based deep reinforcement learning for ransomware detection using ember2018," *Computers & Security*, vol. 131, p. 103211, 2024.

[176] L. Huang, X. Zhang, and W. Wang, "Industrial iot anomaly detection using ddpg in wust-ids2022 dataset," *IEEE Internet of Things Journal*, vol. 10, no. 8, pp. 7154–7166, 2023.

[177] K. Wang, S. Lin, and X. Luo, "Dns tunneling detection via rainbow dqn: Experiments on cic-dns2021," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 3840–3853, 2023.

[178] A. Gupta, R. Singh, and P. Kumar, "Reinforcement learning for adaptive access control: A sac-based approach," in *IEEE DataPort*, 2023.

[179] Y. Zhang, L. Chen, and H. Qian, "Deep reinforcement learning for attack defense in 6g networks," *IEEE Transactions on Network Science and Engineering*, vol. 11, no. 2, pp. 1012–1024, 2024.

[180] A. Singh, P. Sharma, and N. Jain, "Zero-day attack detection in iot using ddpg: An empirical study on bot-iot dataset," *Future Generation Computer Systems*, vol. 147, pp. 54–66, 2023.

[181] C. Gao, H. Liu, and Z. Wang, "Deep reinforcement learning for ddos attack mitigation in software-defined networks," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4123–4136, 2022.

[182] A. Alshamrani and S. Alnabulsi, "Hybrid ids for iot: Integrating reinforcement learning and signature-based detection," *Sensors*, vol. 23, no. 6, p. 2951, 2023.

[183] J. Chen, L. Ma, and Z. Zhao, "Botnet detection using double dqn in ctu-13 dataset," *Future Internet*, vol. 15, no. 7, p. 214, 2023.

[184] Q. Liu, Y. Fang, and L. Zhang, "Malware traffic classification based on drl and cnn," *Journal of Information Security and Applications*, vol. 65, p. 103139, 2022.

[185] J. Xu, Y. Chen, and W. Li, "An adaptive rl-based anomaly detector for industrial iot systems," in *Proceedings of the 2023 IEEE Conference on Communications and Network Security (CNS)*, 2023, pp. 112–119.

[186] S. Ali, H. Abbas, and A. Shafique, "Reinforcement learning-enhanced intrusion detection in ics networks," *IEEE Access*, vol. 12, pp. 22 312–22 325, 2024.

[187] M. Ferrag, L. Maglaras, and D. Fraga, "Deep rl for anomaly detection in cloud environments: A case study," *Journal of Cloud Computing*, vol. 12, no. 1, p. 32, 2023.

[188] A. Mohamed, H. Fawaz, and S. A. Mostafa, "Anomaly detection in smart grids using proximal policy optimization," *IEEE Transactions on Smart Grid*, vol. 13, no. 5, pp. 4169–4177, 2022.

[189] R. Wang, T. Zhang, and Y. Zheng, "Secure access control using rl in 5g networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 2, pp. 1513–1525, 2024.

[190] X. Liu, Z. Chen, and Q. Li, "Zero-day attack detection in iot networks via deep rl," *IEEE Internet of Things Journal*, vol. 10, no. 9, pp. 7516–7527, 2023.

Amir Javadpour holds a Ph.D. in Mathematics/Cybersecurity from Guangzhou University, China. His academic journey is distinguished by numerous publications in highly-ranked journals and prestigious conferences. These works span a diverse range of topics, reflecting his deep expertise in Cybersecurity, Cloud Computing, Software-Defined Networking (SDN), Big Data, Intrusion Detection Systems (IDS), the Internet of Things (IoT), Moving Target Defense (MTD), Machine Learning (ML), Reinforcement Learning, and optimization algorithms. Beyond his publications, he has made substantial contributions as a reviewer and author for leading academic venues, including IEEE Transactions on Cloud Computing, IEEE Transactions on Network Science and Engineering, and ACM Transactions on Internet Technology. His reviewing efforts extend to various reputable journals under Springer and Elsevier. Additionally, he serves as a dedicated Technical Program Committee (TPC) member for several international conferences. Dr. Javadpour actively collaborates internationally, particularly with European consortiums on funded projects such as Inspire-5Gplus (https://www.inspire-5gplus.eu/) and Rigourous (https://rigourous.eu/). These partnerships have resulted in significant contributions to the field, with his work being featured in top-tier journals and conferences, including Globecom, IEEE Transactions on Industrial Informatics (TII), IEEE Transactions on Information Forensics and Security (TIFS), IEEE Transactions on Network and Service Management (TNSM), and ACM Transactions on Sensor Networks (TOSN). In addition to his research and publication efforts, he is deeply committed to mentoring and supervising Master's and Doctoral students. His extensive experience in this area has equipped him with the skills and confidence necessary to lead a research group and conduct independent, high-impact research.

Forough Ja'fari is a Senior Researcher in cybersecurity and computer science. She received her Bachelor's degree from Sharif University of Technology and her Master's degree in Computer Network Engineering from Yazd University, Iran. She is a visiting scholar researcher at Guangzhou University, China. Software-Defined Networking (SDN), Intrusion Detection Systems (IDS), Internet of Things (IoT), Moving Target Defence (MTD), and Machine Learning are some of her research interests. She is currently a Guest Editor (GE) of Cluster Computing (CLUS) Journal, as well as a reviewer for several journals and conferences.

Fatih Turkmen Fatih Turkmen is an Associate Professor at the University of Groningen working on security and privacy. He received his PhD degree from University of Trento (Italy), in 2012. His research interests lie at the security/privacy and machine learning, and involved the frequent use of privacy-enhancing technologies (PET) and empirical/formal security analysis. In recent years, he developed an interest in the privacy of genomic data trying to answer questions like: "How do we ensure the protection of our genomic data which is unique, long-lived and effects also the people related to us?" This includes the study of security/privacy issues and development of mitigation techniques at different levels of the ICT stack.

Chafika Benzaïd is currently a senior research fellow at University of Oulu, Finland. Between Nov. 2018 and Dec. 2021, she was senior researcher at Aalto University. Before that, she worked as an associate professor at University of Sciences and Technology Houari Boumediene (USTHB). She holds Engineer, Magister and "Doctorat ès Sciences" degrees from USTHB. Her research interests lie in the field of 5G/6G, SDN, Network Security, AI Security, and AI/ML for zero-touch security management. She is an ACM professional member.

Tarik Taleb Prof. Tarik Taleb is currently a Chair Professor at Ruhr University Bochum, Bochum, Germany. Prior to that, he was a full professor at the Centre for Wireless Communications (CWC) – Networks and Systems Unit, Faculty of Information Technology and Electrical Engineering, The University of Oulu. Between Oct. 2014 and Dec. 2021, he was a Professor at the School of Electrical Engineering, Aalto University, Finland. Prior to that, he was working as Senior Researcher and 3GPP Standards Expert at NEC Europe Ltd, Heidelberg, Germany. He was then leading the NEC Europe Labs Team working on R&D projects on carrier cloud platforms. Before joining NEC and till Mar. 2009, he worked as assistant professor at the Graduate School of Information Sciences, Tohoku University, Japan, in a lab fully funded by KDDI, the second largest mobile operator in Japan. From Oct. 2005 till Mar. 2006, he worked as a research fellow at the Intelligent Cosmos Research Institute, Sendai, Japan. He received his B. E degree in Information Engineering with distinction, M.Sc. and Ph.D. degrees in Information Sciences from Tohoku Univ., in 2001, 2003, and 2005, respectively. Prof. Taleb's research interests lie in the field of telco cloud, network softwarization and network slicing, AI-based software defined security, immersive communications, mobile multimedia streaming, next generation mobile networking. Prof. Taleb was also directly engaged in the development and standardization of the Evolved Packet System as a member of 3GPP's System Architecture working group 2. Prof. Taleb served on the IEEE Communications Society Standardization Program Development Board. As an attempt to bridge the gap between academia and industry, Prof. Taleb founded the "IEEE Workshop on Telecommunications Standards: from Research to Standards", a successful event that got awarded the "best workshop award" by IEEE Communication Society (ComSoC). Based on the success of this workshop, Prof. Taleb also founded and served as the steering committee chair of the IEEE Conf. on Standards for Communications and Networking. Prof. Taleb served as the general chair of the 2019 edition of the IEEE Wireless Communications and Networking Conference (WCNC'19) held in Marrakech, Morocco. He was the guest editor in chief of the IEEE JSAC Series on Network Softwarization and Enablers. He was on the editorial board of the IEEE Transactions on Wireless Communications, IEEE Wireless Communications Magazine, IEEE Journal on Internet of Things, IEEE Transactions on Vehicular Technology, IEEE Communications Surveys & Tutorials, and a number of Wiley journals. Till Dec. 2016, he served as chair of the Wireless Communications Technical Committee, the largest in IEEE ComSoC. He also served as Vice Chair of the Satellite and Space Communications Technical Committee of IEEE ComSoc (2006 - 2010). Prof. Taleb is the recipient of the 2021 IEEE ComSoc Wireless Communications Technical Committee Recognition Award (Dec. 2021), the 2017 IEEE ComSoc Communications Software Technical Achievement Award (Dec. 2017) for his outstanding contributions to network softwarization. He is also the (co-) recipient of the 2017 IEEE Communications Society Fred W. Ellersick Prize (May 2017), the 2009 IEEE ComSoc Asia-Pacific Best Young Researcher award (Jun. 2009), the 2008 TELECOM System Technology Award from the Telecommunications Advancement Foundation (Mar. 2008), the 2007 Funai Foundation Science Promotion Award (Apr. 2007), the 2006 IEEE Computer Society Japan Chapter Young Author Award (Dec. 2006), the Niwa Yasujirou Memorial Award (Feb. 2005), and the Young Researcher's Encouragement Award from the Japan chapter of the IEEE Vehicular Technology Society (VTS) (Oct. 2003). Some of Prof. Taleb's research work have been also awarded best paper awards at prestigious IEEE-flagged conferences.