Intelligent Systems with Applications 26 (2025) 200495

Contents lists available at ScienceDirect

Intelligent Systems with Applications

journal homepage: www.journals.elsevier.com/intelligent-systems-with-applications

Review

Multi-Agent Reinforcement Learning for Cybersecurity: Classification and survey

Salvo Finistrella *, Stefano Mariani, Franco Zambonelli

Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, Via Amendola, 2, Reggio Emilia, 42122, RE, Italy

ARTICLE INFO

Keywords:
Reinforcement learning
Cybersecurity
Multi-agent system
Attack mitigation
Software Defined Networking
Intrusion detection system

ABSTRACT

In the face of a rapidly evolving threat landscape, traditional cybersecurity measures – such as signature-based detection and static rules on firewalls, intrusion detection systems (IDS) and antivirus software – often lag behind sophisticated cyber attacks. Through a review of existing literature, we examine the shortcomings of traditional cybersecurity methods and how these can be surpassed with the application of Reinforcement Learning (RL) based methods. This study classifies RL-based approaches to cybersecurity, aimed at enhancing detection, mitigation and response to cyber attacks, along two orthogonal dimensions: the RL Frameworks used (e.g. single-agent vs. multi-agent) and the network configuration where they are deployed (e.g. host-based, or network-based cybersecurity). The goal is that of aiding researchers and practitioners interested in the field to quickly understand what are the opportunities for RL-based cybersecurity depending on the network environment to be protected and point them to the representative articles in the field. Finally, we emphasize the importance of further research and development to address challenges such as computational complexity, generalization and data quality.

1. Introduction

On a global scale, projections indicate that the cost of cybercrime will surpass 8 trillion dollars, cementing its status as the world's third-largest and most rapidly expanding economy (Morgan, 2022). Such cost includes damage and destruction of data, stolen money, lost productivity, theft of intellectual property as well as personal and financial data, fraud, post-attack disruption, forensic investigation and reputational harm. Traditional security systems are often *reactive* and struggle to keep up with the rapidly evolving threat landscape. Moreover, they require constant human intervention and/or supervision, that delays the response process.

In recent years, Artificial Intelligence (AI) and Machine Learning (ML) have revolutionized cybersecurity by providing intelligent and automated methods to detect and mitigate threats. Techniques such as supervised and unsupervised learning have been applied to anomaly detection, malware classification and network intrusion detection systems (Martínez Torres, Iglesias Comesaña, & García-Nieto, 2019; Sarker et al., 2020; Sharafaldin, Lashkari, & Ghorbani, 2018; Xin et al., 2018). Swarm intelligence and evolutionary computation methods have also emerged as promising tools, taking advantage of collective behaviors to optimize defense mechanisms (He et al., 2016; Mishra, Sagban, Yakoob, & Gandhi, 2021). However, while these methods excel at tasks for stable scenarios, they often lack the adaptability required to handle dynamically evolving threats in real time, as better discussed in Section 2.

In this context, *Reinforcement Learning* (RL) and, in particular, *Multi-agent Reinforcement Learning* (MARL), stand out by providing *dynamic* and *adaptive* capabilities to strengthen cybersecurity defenses, creating solutions that learn and evolve in response to threats (Adawadkar & Kulkarni, 2022; Cengiz & Gök, 2023; Ozkan-Okay, Akin, Aslan, Kosunalp, Iliev, Stoyanov, & Beloev, 2024; Uprety & Rawat, 2021).

RL is a field of machine learning where an active entity, called *agent*, learns to make decisions by interacting with an *environment*. Through trial and error, the agent receives *rewards* for taking actions that advance it toward a desired goal. By maximizing accumulated rewards, the agent progressively learns which actions lead to favorable outcomes and refines its behavioral policy (policy, for short) accordingly (Connell & Sridhar Mahadevan, 1999).

RL enables security systems to actively learn from experience and autonomously adapt their strategies in response to changing threats. By automating decision-making, in fact, RL reduces the need for human intervention, minimizing response times and mitigating the risk of human errors. Furthermore, RL continuous learning capability allows systems to stay updated with emerging threats, making it highly effective against novel attacks (especially, zero-day attacks Guo, 2023).

* Corresponding author.
E-mail addresses: salvo.finistrella@unimore.it (S. Finistrella), stefano.mariani@unimore.it (S. Mariani), franco.zambonelli@unimore.it (F. Zambonelli).

https://doi.org/10.1016/j.iswa.2025.200495
Received 30 September 2024; Received in revised form 24 January 2025; Accepted 14 February 2025
Available online 25 February 2025

Building on our previous work (Finistrella, Mariani, & Zambonelli, 2024), this article presents a comprehensive classification of RL Frameworks applied to cybersecurity, categorizing them based on the network environment (e.g. traditional vs. software defined) and the learning setting (e.g. single-agent vs. multi-agent, with particular attention to MARL). Our classification covers RL Frameworks there including deep learning and adversarial settings, exploring applications across host-based, network-based and centralized network-based configurations using Software Defined Networking (SDN). Differently from existing surveys, our work not only aims at guiding researchers and practitioners to quickly understand at a glance how RL can be effectively applied in different network contexts, but also zooms-in into specific RL approaches for each identified category and highlights areas where further research is needed. Moreover, we include a deeper analysis of adversarial RL, focusing on its significance in addressing evolving threats, a detailed discussion on the role of simulators and datasets in the field, and an overview of the practical challenges in applying RL to cybersecurity, such as scalability and generalization.

The remainder of the paper is organized as follows: Section 2 discusses the limitations of traditional cybersecurity approaches, thus the need for RL; Section 3 introduces the key RL concepts and discusses how they can be instantiated specifically in a cybersecurity environment; Section 4 introduces the two dimensions of classification that we adopt to categorize the surveyed papers (network configuration and architecture of the RL Frameworks); Section 5 describes the surveyed paper organized according to the proposed classification; Section 6 discusses the current issues of RL-based cybersecurity solutions, highlighting challenges and future research directions; Section 7 concludes the paper.

2. Motivation & background

The rapid evolution of cyber threats in today's digital landscape reveals the limitations of traditional cybersecurity measures. These struggle to keep pace with novel and sophisticated attack vectors. Below are the key challenges faced by conventional approaches:

- **Static Defences:** Traditional systems rely heavily on fixed rules and predefined attack signatures, which limit their effectiveness in dealing with changing threats and vulnerabilities (Hu, Chen, Zhu, & Liu, 2019).
- **Lack of Contextual Understanding:** Standard approaches typically cannot understand the full context of a cyberattack, such as the peculiar network environment, the specific strategies employed by attackers and the overall highly dynamic and unpredictable nature of the attacks. This often leads to incomplete threat detection and ineffective responses to emerging, complex attacks (Macas, Wu, & Fuertes, 2022).
- **Limited Scalability:** As data volumes and the diversity of threats increase, traditional cybersecurity systems can become simply overwhelmed, rendering them inefficient and slow in processing security events (Nguyen & Reddi, 2023).
- **Inadequate Response Times:** Human intervention is often too heavily required in traditional systems, leading to delays that are detrimental during time-sensitive security incidents (Nguyen & Reddi, 2023).

Over the years, researchers have explored various AI and ML-based approaches to address these challenges. Supervised ML techniques, for instance, exploit labeled datasets to detect malicious patterns, whereas unsupervised methods like clustering are mostly used for anomaly detection (Yaseen, 2023). While effective, these methods often require extensive feature engineering and struggle to adapt to new attack vectors without retraining. Swarm intelligence and evolutionary computation techniques have also been applied to optimize intrusion detection and other defensive mechanisms by applying the principles of natural selection and collective behavior (He et al., 2016; Mishra et al., 2021). However, these approaches often lack scalability and real-time adaptability when deployed in highly dynamic environments. Furthermore, as highlighted in Mishra et al. (2021), swarm intelligence methods, while powerful in optimizing specific tasks, suffer from high computational costs and convergence issues in complex, large-scale networks. Similarly, evolutionary computation techniques are limited by their reliance on large search spaces and are often computationally expensive, making them less practical for real-time applications in IoT-enabled cyber–physical systems (He et al., 2016).

These limitations highlight the need for a more adaptive, scalable and responsive approach. RL provides a significant advantage by enabling systems to *autonomously* adapt to emerging threats without predefined labels or exhaustive feature engineering. RL's trial-and-error learning process allows for continuous adaptation, making it particularly effective for addressing zero-day attacks and other unknown vulnerabilities (Guo, 2023).

By integrating RL into cybersecurity, several limitations of traditional approaches are addressed. Some of the key beneficial properties of RL-based systems include:

- **Dynamic Threat Detection:** RL algorithms continuously adapt by interacting with their environment, enhancing their ability to detect both known and unknown threats in real-time. This capability tackles the rigidity of static defences and addresses the scalability issue (Hu et al., 2019).
- **Real-Time Threat Analysis:** RL improves real-time analysis by factoring in diverse contextual information—such as network traffic patterns, system anomalies and user behavior. This significantly reduces response time while offering a deeper understanding of the attack context, surpassing traditional systems limited by manual response times and attack signatures (Nguyen, Nguyen, Nguyen, & Nahavandi, 2020; Nguyen & Reddi, 2023).
- **Enhanced Decision-Making:** RL ability to learn from experience allows it to make progressively better decisions, improving the detection of anomalies and the mitigation of potential threats (Fragkos, Johnson, & Tsiropoulou, 2022).
- **Automation and Efficiency:** By automating and optimizing repetitive tasks like traffic monitoring and malware detection, RL systems are highly scalable, enabling them to handle vast amounts of data and diverse attack vectors without requiring human intervention (Gulmez & Angin, 2020; Sewak, Sahay, & Rathore, 2022).
- **Minimization of False Positives:** RL models learn to distinguish between routine and malicious activities over time, which reduces false positives and helps security teams focus on legitimate threats (Adawadkar & Kulkarni, 2022).

The benefits offered by RL systems underscore their potential to revolutionize cybersecurity, making them essential in addressing the evolving threat landscape of today's digital world.

Table 1 summarizes how these RL benefits address the core limitations of traditional cybersecurity approaches. Each row represents a key capability of RL that enhances security measures, while the columns correspond to the specific challenges of static defences, lack of context, scalability and response times.

Despite increasing interest in RL applications, the research landscape remains fragmented, with several surveys focusing on narrow domains like IoT security or intrusion detection, or directly delving into specific RL algorithms without clarifying "the big picture" (Adawadkar & Kulkarni, 2022; Cengiz & Gök, 2023; Dixit & Silakari, 2021; Nguyen & Reddi, 2023; Uprety & Rawat, 2021).

Our survey aims to fill this gap by providing a *classification framework* that categorizes RL approaches to cyber defense according to different network environments and RL Frameworks. Instead of the

Table 1
Limitations of traditional approaches (columns) and how RL helps improving (rows).

	Adaptive defences	Context understanding	Scalability	Response times
Dynamic detection	✓		✓	
Real-time analysis	✓	✓		✓
Enhanced decision-making		✓		
Efficiency through automation			✓	✓
Minimising false positives				✓

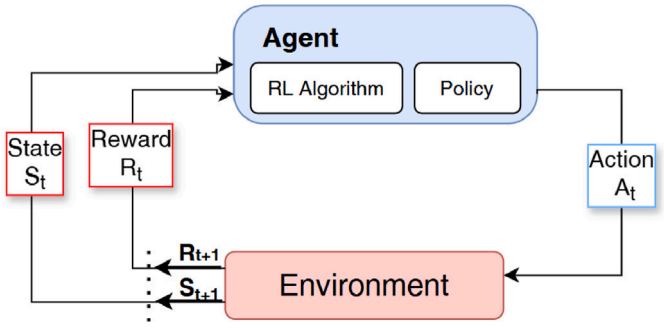

Fig. 1. Core components of the reinforcement learning loop.

technical details of individual RL algorithms, we focus on helping researchers and practitioners select the appropriate RL Frameworks based on the network configuration where RL will be applied—whether it is host-based, or network-based (see Tables 3–7). This structured approach bridges the gap between theoretical discussions and real-world deployment, offering practical insights into how RL can be tailored to specific security needs and networks without requiring extensive algorithmic expertise.

Furthermore, we offer a critical analysis of adversarial RL Frameworks, which are crucial in addressing the adaptive nature of modern cyber threats. We also emphasize the importance of practical implementation, including the use of simulation software and addressing challenges related to scalability and coordination in multi-agent systems. By focusing on these aspects, our survey goes beyond previous reviews, providing a deeper understanding of RL's applicability and identifying areas for future research.

3. Reinforcement learning and cybersecurity

RL represents a powerful learning paradigm in the domain of artificial intelligence, enabling agents to learn optimal behavior through repeated interaction with their environment, without any supervision. Mimicking the trial-and-error learning process observed in humans and animals, RL algorithms iteratively explore and exploit their environment to maximize cumulative rewards. RL has applications in diverse fields where systems must autonomously adapt to uncertain and dynamic environments, such as robotics (Kober, Bagnell, & Peters, 2013), gaming (Shao, Tang, Zhu, Li, & Zhao, 2019), finance (Fischer, 2018) and healthcare (Yu, Liu, Nemati, & Yin, 2021).

In RL, an agent interacts with an environment to achieve a goal. The **agent** takes **actions** based on the current state of the environment and the **environment** responds by providing feedback in the form of **rewards** (Fig. 1). The agent's objective is to maximize the cumulative reward over time by learning which actions lead to the most favorable outcomes. To achieve this, the agent can use various algorithms to create and refine its **policy**, which determines its actions at any given state. These algorithms range from basic approaches, such as Q-learning and SARSA, to advanced methods like deep reinforcement learning and policy gradient techniques, each tailored to different types of problems and environments This process is iterative: as the agent continues to explore different actions, experience rewards and apply its chosen algorithm, it progressively improves its policy to make better decisions in the future. Unlike supervised learning, where the model learns from a fixed labeled dataset, RL involves learning from continuous interactions with a dynamic environment.

3.1. RL in cybersecurity

In the cybersecurity domain, the RL agents are tasked with defending a system against threat actors, which can be actual human/software attackers or emulated through datasets and simulation frameworks. These threat actors strategically exploit the environment to find vulnerabilities. In scenarios involving multiple agents, as depicted in Fig. 2, **MARL** allows agents to collaborate, sharing observations and coordinating actions to address threats more effectively.

- **Environment state and observations**: RL algorithms rely on observing the state of the environment to make decisions. Such observations may be *perfect* (the true environment state can be observed without errors), *partial* (only part of the state can be perceived) and/or *noisy* (observations may include errors). As depicted in Fig. 2, in cybersecurity the environment is the network infrastructure, within which several appliances generate the data constituting the state: firewalls, Intrusion Detection Systems (IDSs), Intrusion Prevention Systems (IPSs), proxy servers, sniffers, Operating Systems and other software. This entails monitoring heterogeneous data such as network traffic patterns, system logs, software configurations and user behavior. The generated observations serve as inputs for the agents to learn.
- **Action selection**: after observing the state, the RL agent selects actions based on its learned policy or exploration strategy, there including triggering alarms, deploying patches, updating security configurations, isolating compromised systems, alerting security personnel, block services and dropping packets. The RL environment evaluates the efficacy of these actions by assessing whether the state of security has improved or deteriorated. *Rewards* or penalties can then be issued accordingly (see below).
- **Reward mechanisms**: *rewards* provide feedback to the agent, indicating the efficacy of its actions, hence are crucial to let the agent understand how to behave to advance towards its goal. In cybersecurity, their primary goal is to incentivize actions leading to successful detection and mitigation of threats. To achieve this, reward design could involve awarding accurate identification and swift response to threats while penalizing false positives and negatives. The specifics of the reward function formulation vary, contingent upon the objectives of the security system at hand. Section 5 provides practical examples of rewards in cybersecurity for each class of approaches.
- **Policy optimization**: RL agents refine their decision-making through *trial and error*, continuously learning to maximize long-term rewards. This process involves balancing *exploration* – testing new actions to understand their impact—and *exploitation* – choosing actions that have previously yielded the highest rewards. Over time, agents improve their strategy by learning from past outcomes, becoming more effective in dynamic environments. Guided by overarching security goals, RL agents prioritize actions that reduce *risk* and protect critical *assets*, enhancing the overall security posture.

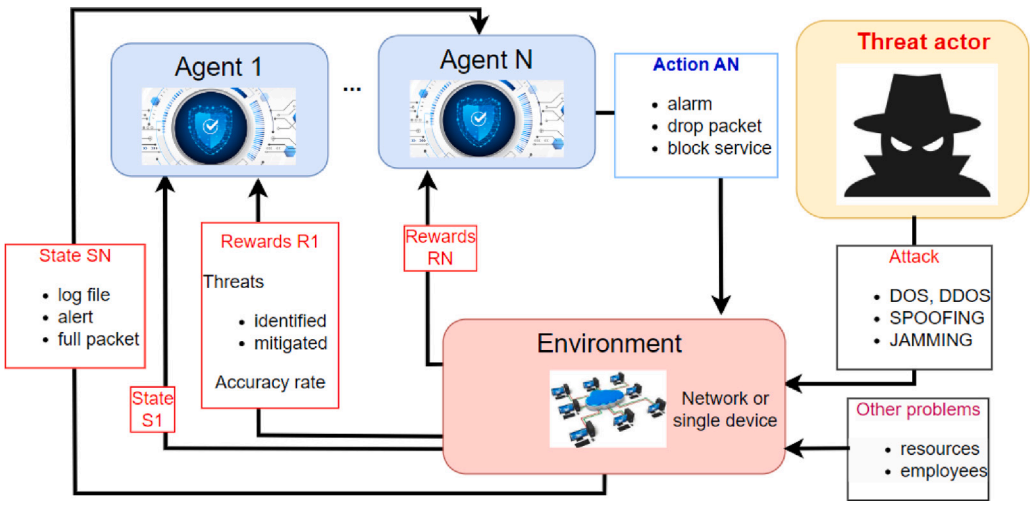

Fig. 2. A typical MARL environment within the domain of cybersecurity. One or multiple agents interact with a network environment with several appliances available but also susceptible to attacks. A threat actor is also present, in the form of one or multiple active human/software attackers, cyber threat simulation software, or dataset of past attacks.

In the cybersecurity domain, the RL agents are tasked with defending a system against threat actors, that can be either actual human/software attackers or emulated through datasets and simulation frameworks. These threat actors strategically exploit the environment to find vulnerabilities and additional challenges may arise from authorized resources or employees attacked to get access to other parts of the system.

3.2. Related surveys

To investigate how RL is applied in the realm of cybersecurity and define our own classification, we started from a selection of key surveys that offered a broad overview of the field. These works span various application areas, including general cybersecurity and specific domains like the IoT. This way, we were able to extract insights into the primary trends, challenges and advances in the intersection between RL and cybersecurity, allowing us to identify a set of core studies that reflect distinct approaches and innovations in applying RL to enhance security measures.

We considered a broad range of surveys in the field, allowing us to compare and contrast different perspectives. Among these, we selected five key surveys (Adawadkar & Kulkarni, 2022; Cengiz & Gök, 2023; Dixit & Silakari, 2021; Nguyen & Reddi, 2023; Uprety & Rawat, 2021) that offered a solid foundation for our analysis. From this foundation, we applied a snowballing technique to identify a selection of representative articles that would faithfully and clearly capture important research threads within our proposed classification framework.

In the first, Uprety and Rawat (2021) provide a comprehensive overview of various *cyberattacks that target IoT systems*, essential for understanding the threat landscape that RL-based security solutions need to address. By categorizing these attacks, the survey offers a clear framework for identifying and analyzing potential vulnerabilities in IoT networks. Additionally, this survey explores different RL and deep RL-based security solutions designed to mitigate the attacks, emphasizing RL's ability to dynamically adapt to new threats. Its tabular summaries of recent attacks and countermeasures provide a quick reference for the effectiveness and application of different RL Frameworks and algorithms, making it easier to identify which methods might be most applicable or beneficial for specific IoT security challenges. On the contrary, we focus on which RL Frameworks are best suited for which network configuration and only then describe the kind of attacks. This allows readers to quickly select the relevant papers best aligned with their actual target network environment.

Adawadkar and Kulkarni (2022), instead, offer a comprehensive review of RL Frameworks applied to various cybersecurity domains. This survey focuses on IDS, IPS, IoT and Identity and Access Management (IAM). The paper *identifies key evaluation parameters* such as detection rate, precision, and accuracy, which are essential for comparing RL-based algorithms. It highlights the current state of RL applications, datasets used and existing gaps in the literature. Unlike this survey, our review is focussed on how different RL Frameworks perform specifically in diverse network contexts, providing a practical orientation for deploying RL strategies based on the characteristics of the deployment environment.

Cengiz and Gök (2023) cover key areas such as penetration testing, IDS, and cyberattacks. The detailed analysis of various RL Frameworks and their effectiveness in enhancing cybersecurity measures offers valuable insights and evidence to support research. Additionally, the survey highlights the evolving nature of cybersecurity threats and how RL can be applied to mitigate these risks, thus emphasizing the relevance and potential of RL in fortifying defenses against emerging cyber threats. While this survey provides a broad analysis across multiple RL applications, our work differentiates itself by offering a more granular classification that directly links RL Frameworks to specific network configurations, including host-based, network-based and SDN-based environments. This added focus on network configurations not only supports a more tailored application of RL Frameworks but also provides practical guidelines for selecting the most appropriate methods based on the operational context. Moreover, our survey delves deeper into the role of multi-agent systems and their applicability in complex, distributed network environments, a perspective that is only briefly touched upon in Cengiz and Gök (2023)'s work.

Nguyen and Reddi (2023) explore the intersection of *Deep RL* (DRL) techniques and the evolving landscape of cyber threats, showing how these techniques offer adaptive, scalable and robust solutions to counter increasingly sophisticated cyberattacks. They underlines the limitations of traditional methods in providing real-time response capabilities in cybersecurity. In contrast, our review not only discusses the adaptability of DRL but also systematically categorizes these techniques based on their suitability for different network architectures, providing a clear framework for their implementation.

Finally, Dixit and Silakari (2021) highlight how RL Frameworks enhance the detection and mitigation of cybersecurity threats such as malware, phishing, Denial of Service (DoS) attacks and more. Understanding these applications helps in developing more robust and adaptive cybersecurity measures, using the strengths of RL for continuous learning and adaptation in dynamic threat environments. While this

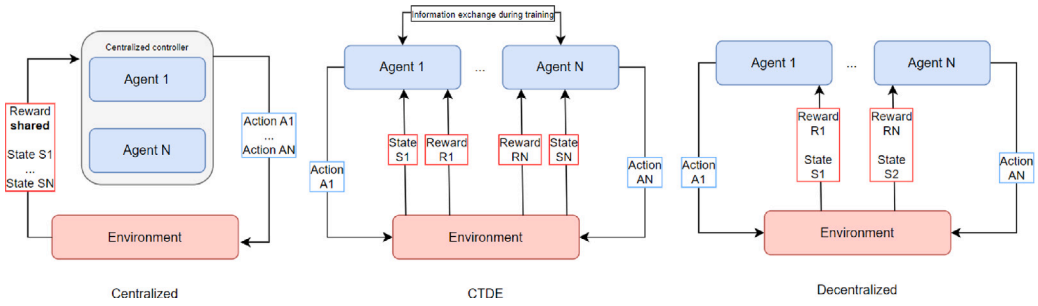

Fig. 3. Training schemes in the multi-agent learning setting.

survey offers a broad overview of RL applications in cybersecurity, our review takes a focused approach by mapping RL methods to network types, thereby offering insights into the most effective use cases for RL across different cybersecurity scenarios.

Our review builds on these surveys by organizing selected representative research papers into a structured classification that facilitates understanding of the key methodologies, architectures and application areas within RL for cybersecurity. In doing so, we aim to provide a "bird-eye-view" over (deep) RL-based cybersecurity for researchers and practitioners seeking to navigate this fast-evolving field.

4. Classification

Differently from previous surveys, we formulate a novel classification encompassing two key dimensions: *(i)* the RL Frameworks employed and *(ii)* the network configuration adopted for cybersecurity.

4.1. RL frameworks dimension

The field of RL is grown to a vast discipline with plenty of approaches available, that can be divided into families according to several characteristics such as the number of learning agents, the algorithm adopted, etc. Here we provide a bird-eye-view over RL Frameworks by categorizing them into 3 macro classes based on both the number of learning agents and their relationship with each other:

- **Single-agent**. In single-agent systems, there is only *one learning agent* in the environment. In the case of cybersecurity, this means that this agent learns to make decisions and take actions based solely on its own observations, controlling the entire system.
- **Multi-agent**. In Multi-Agent RL (MARL), *multiple agents learn concurrently* as part of a multi-agent system (MAS). The way they share information or consider each other activities defines the specific learning architecture (see Fig. 3 and Table 2, column Multi-agent), that we overview in the following.

As far as MARL is concerned, several different solutions can be conceived.

In *centralized* MARL, there is a *central controller or coordinator* responsible for learning a *single decision-making policy* for all agents based on the information collected from each agent. This central controller has access to all the data and actions of the agents, allowing it to learn a single, global policy that is then distributed to all the agents. As a result, the *agents are not independently learning*; instead, they merely execute the decisions dictated by the central controller. This approach simplifies learning since the controller has access to the complete state and action space, but it does not scale well in environments where agents must make decisions rapidly, communicate in real-time, or adapt to changes locally. Moreover, it may not be feasible in situations where agents are geographically distributed or operate under privacy constraints that prevent centralized data aggregation.

The *Centralized Training Decentralized Execution* (CTDE) architecture has become the most popular approach in MARL. The core idea is to *separate the training and execution phases*, similar to the state-of-the-art practice in supervised machine learning. During training, a central controller can utilize global information from all agents, or agents can communicate freely to optimize learning. This access to global state information or inter-agent communication enables the learning of a robust policy that accounts for the interactions and dependencies between agents.

However, *during execution, the central coordinator and/or the possibility of communicating is removed* and agents must operate independently. They make decisions based on the policy learned during the training phase but conditioned on their own local observations. This approach allows for scalability and robustness in environments where agents may not have access to global information or direct communication with other agents. It combines the best of both worlds: the advantages of centralized learning during training and the flexibility and adaptability of decentralized decision-making during execution. This makes CTDE particularly suited for real-world applications like autonomous driving, distributed sensor networks and robotic swarm control, where centralized coordination is impractical during operation, but attainable during training in simulated environments.

Finally, in *decentralized* MARL, each agent makes its own decisions, *without a central controller*—not in training, nor in execution. Agents in decentralized systems can have no, or limited access to information about other agents (e.g. their observations, their current policies, their last actions, etc.). Hence, they typically use at most local information and possibly communicate solely with nearby agents to learn and make decisions. Whatever the case, *each agents learns its own policy*.

Another relevant MARL learning setting that we describe separately due its peculiar, *competitive setting*, is:

- **Adversarial Multiagent**. In this setup, agents operate in a competitive environment where each agent's objectives are directly opposed to those of other agents. Typically, there are at least two agents involved: one or more *adversarial agents*, often representing attackers and one or more *defender agents* working to counteract those attacks. Both sides engage in strategic interactions, where *both agents must learn* to anticipate and react to the actions of others to achieve their own objectives. Importantly, both sets of agents are learning concurrently, adapting their strategies based on the evolving tactics of their opponents. This learning dynamic fosters a constantly shifting competitive landscape, where agents not only learn from their own experiences but also from predicting the moves of their adversaries.

4.2. Network configuration dimension

The other dimension we consider is the *network configuration* of the cybersecurity environment.

- **Host-based cybersecurity**. The defence system is focused on *protecting individual devices* (hosts) such as computers, servers,

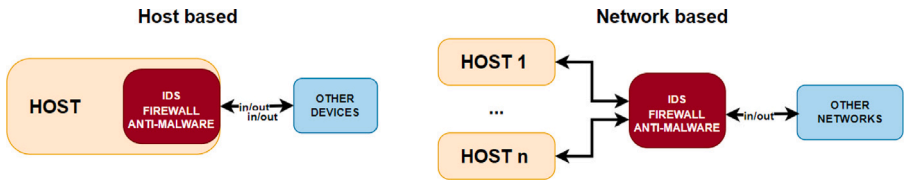

Fig. 4. The figure illustrates the two configurations for deploying cybersecurity measures: *host-based* and *network-based* security. In reality, these two configurations can coexist within a single system to provide comprehensive protection. Despite the complementary nature of these configurations, RL Frameworks tend to specialize in one configuration at a time, targeting either host-level threats or network-level vulnerabilities. This specialization allows RL models to optimize their learning and actions based on the specific challenges inherent to each domain.

mobile devices and endpoints. As such, security software is directly installed on these devices and operates locally. This software may include antivirus programs, firewalls, intrusion detection/prevention systems (IDS/IPS) (Denning, 1987) and endpoint protection platforms (EPP). Host-based cybersecurity measures are essential for safeguarding against threats like malware, unauthorized access and data breaches that may target specific devices (see Fig. 4).

- **Network-based cybersecurity**. Network-based security expands the focus from individual devices to *securing the communication pathways* between devices and subsystems within a network. While host-based systems protect single endpoints, network-based security solutions deal with threats that spread across the entire network, such as malware *propagation* and Distributed Denial of Service attacks. These solutions include firewalls, intrusion detection/prevention systems (IDS/IPS), virtual private networks (VPNs) and network access control (NAC) systems, which are implemented at the network level (e.g., onto routers, switches and gateways) to detect and prevent unauthorized access and malicious activities across the network (see Fig. 4). Network-based security addresses all the threats that host-based systems target, but with the added complexity of managing network-wide vulnerabilities and the *replication and spread of attacks* between hosts.
- **Network-based with SDN**.
  This configuration combines network-based cybersecurity measures with Software-Defined Networking (SDN, see Fig. 5) technology. SDN (Shin, Xu, Hong, & Gu, 2016) is an approach to networking that *separates the control plane from the data plane*, allowing for centralized management and programmability of network resources. The control plane governs everything related to forwarding data packets, or how data is sent from a source to a destination. The control plane is responsible for various functions and processes such as routing table creation, maintenance, forwarding, establishing network policies, managing routing protocols and creating the network topology. The data plane performs tasks like receiving and inspecting the packets, forwarding them to the correct destination, queuing network packets during network congestion and ensuring the safe delivery of packets to the correct destination. This enables more dynamic and granular control over network traffic, enhancing scalability, optimizing performance and improving security. SDN-based security solutions include centralized firewall management, dynamic access control policies and real-time threat intelligence integration, among others.

4.3. Intersection of RL frameworks and network configurations

The intersection of RL Frameworks and network configurations plays a crucial role in designing effective cybersecurity defense mechanisms. The classification of network environments into host-based, network-based and network-based with SDN offers flexibility for the design of a novel system, or introduces valuable constraints if the system at hand is a legacy system to improve with RL.

For instance, in highly sensitive environments like financial institutions, a centralized network configuration may be preferred for its ability to provide comprehensive monitoring and control. However, this setup can become a single point of failure if compromised. On the other hand, in large-scale distributed networks, such as those used by multinational corporations, a decentralized or network-based configuration might be necessary to ensure scalability and resilience. This approach reduces the risks associated with a single point of failure but introduces increased complexity in coordinating defense strategies across multiple nodes. Similarly, in environments using Software-Defined Networking (SDN), centralized control enhances network flexibility and allows for rapid adaptation to new threats, yet the SDN controller itself becomes a potential vulnerability if targeted by attackers.

In parallel, selecting the appropriate RL Frameworks is influenced by both the network configuration and the nature of the cybersecurity tasks. For instance, in host-based environments, a single-agent RL approach might be more practical due to its simplicity and reduced computational overhead. In these cases, the agent can efficiently learn and optimize policies for securing individual hosts. Conversely, for network-based environments, especially in distributed networks, multi-agent RL Frameworks may be more suitable. Multi-agent systems can coordinate across the network, with each agent specializing in different security tasks, such as managing firewalls, intrusion detection systems, or data traffic control. In more complex network-based configurations with SDN, hybrid approaches may be employed, where single-agent systems handle centralized tasks while multi-agent systems manage distributed, localized security challenges.

Furthermore, the selection of RL Frameworks also depends on the type of cybersecurity threats. In dynamic and adaptive environments where adversaries continuously evolve their strategies, adversarial RL Frameworks may be necessary to keep defense mechanisms robust. In contrast, environments facing more predictable and uniform threats can benefit from simpler RL Frameworks, which allow for faster development and deployment of effective security solutions without significant computational overhead.

In summary, understanding the characteristics of both the network configuration – whether host-based, network-based, or network-based with SDN – and the cybersecurity tasks at hand is essential for selecting the most suitable RL Frameworks. This decision directly influences the complexity, scalability and effectiveness of the security system, allowing system administrators to tailor defense mechanisms that best fit their operational requirements and threat landscape.

5. Survey

By organizing our analysis along the above described dimensions, our survey aims at providing a comprehensive overview of the strengths, limitations and potential applications of diverse RL approaches within diverse network settings. This way, we can construct a "bird-eye-view" portrait of the current state-of-the-art in RL applications for cybersecurity (depicted in Table 2) highlighting emerging trends and challenges. Only then the specific RL algorithm can be selected. In fact, while our primary aim is not to delve into the specifics of RL algorithms, we recognize that choosing an appropriate

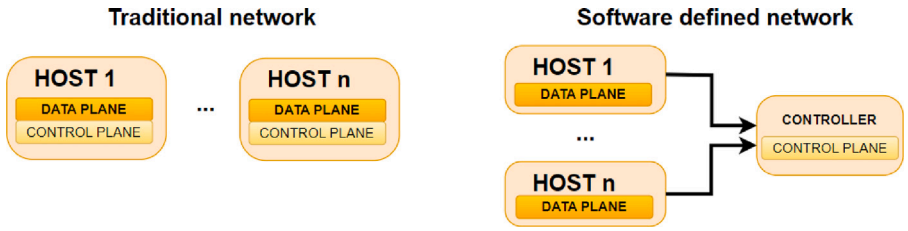

Fig. 5. Traditional network vs. Software defined network.

Table 2
Classification: environment configurations (columns), RL Frameworks (rows).[c] = centralized,[ctde] = ctde,[d] = decentralized.

	Single agent	Multi-agent	Adversarial
Host based	Elnaggar and Bezzo (2018), Feng, Li, and Nguyen (2020), Liu, Xiao, Liu, and Zhao (2017), Oh and Iyengar (2019), Xiao et al. (2018), Xu (2010), Xu and Luo (2007) and Xu and Xie (2005)		
Network based with SDN	Akbari, Tahoun, Salahuddin, Limam, and Boutaba (2020), Guo, Lin, Li, and Peng (2020), Kim, Yoon, and Lim (2021), Liu, Dong, Ota, Li, and Wu (2018), Phan et al. (2019, 2020), Zhang, Qiu et al. (2023) and Zolotukhin, Kumar, and Hämäläinen (2020)	Dake, Gadze, Klogo, and Nunoo-Mensah (2021)[ctde] Janakiraman and Deva Priya (2023)[c] Rezapour and Tzeng (2022)[c] Simpson, Rogers, and Pezaros (2020)[d] Yungaicela-Naula, Vargas-Rosales, Pérez-Díaz, and Carrera (2022)[c]	Chowdhary et al. (2021) and Novaes, Carvalho, Lloret, and Proença (2021)
Network based		Bhagyashree Deokar (2012)[d] Bhosale, Mahajan, and Kulkarni (2014)[d] Dasgupta, Ghosh, and Rahman (2022)[c] Malialis and Kudenko (2015)[d] Shamshirband, Patel, Anuar, Kiah, and Abraham (2014)[d] Sethi, Madhav, Kumar, and Bera (2021)[d] Veluchamy and Kathavarayan (2022)[c]	Anderson, Kharkar, Filar, Evans, and Roth (2018), Apruzzese, Andreolini, Marchetti, andrea, and Colajanni (2020), Caminero, Lopez-Martin, and Carro (2019), Elderman, Pater, Thie, Drugan, and Wiering (2017), Kunz, Fisher, Novara-Gsell, Nguyen, and Li (2022), Piplai et al. (2022), Shashkov, Hemberg, Tulla, and O'Reilly (2023) and Turner, Hemberg, and O'Reilly (2022)

algorithm is vital for practical implementations. Accordingly, at the end of each subsection describing RL solutions at the crossroads of the two dimensions, we discuss the specific RL algorithms employed, also conveniently reported in summary tables.

In our research, we have not encountered significant studies in certain areas, which could be attributed to several factors. For example, single-agent network-based security might be underexplored due to the complexity and scale of managing an entire network's security configurations with a single agent. Veluchamy and Kathavarayan (2022), for instance, studied this combination but ultimately preferred the use of multi-agent strategies for network defense. This technique may not effectively address the diverse and dynamic nature of network threats, necessitating more collaborative multi-agent strategies.

On the contrary, host-based security often benefits more from single-agent technique that can comprehensively manage security settings without the overhead of coordination among multiple agents. Also adversarial host-based security appears to be less studied, which can be due to the challenges in simulating realistic adversarial scenarios at the host level. Host-based systems often focus on direct and immediate threat mitigation, whereas adversarial strategies are more complex and typically explored in network-wide contexts where the interplay between attacking and defending agents can be more dynamically modeled and analyzed.

In the following, we dive deeper into those categories that are most represented.

5.1. Single-agent RL in host-based security

This category represents those solutions that are technically easier to set up (see Table 3 at the end of this subsection): a single learning agent learns based on the inputs coming from all the host devices and controls all the security measures therein installed.

Liu et al. (2017) present a RL-based approach to enhance the security of wireless networks by mitigating spoofing attacks. The receiver (Bob) acts as an agent using the Q-learning algorithm to make decisions about authenticating packets. Bob's state represents his current knowledge of the channel conditions and historical authentication results. Bob's **action** dynamically adjusts a threshold using Q-learning to optimize the trade-off between false alarm rates (rejecting legitimate signals) and miss rates (accepting spoofed signals). The selected threshold directly influences Bob's decision, determining whether the observed signal characteristics fall within the expected range for legitimate signals or if they indicate a potential spoofing attempt. The threshold is influenced by comparing MAC addresses and RSSIs (measurement of the power level that is received by an antenna from a wireless device).

Bob **observes** the packet's physical-layer characteristics, such as signal strength and channel properties. The **reward** function (r_t) provides feedback based on the accuracy of Bob's authentication decisions, rewarding correct identifications and penalizing false alarms and missed detections, as follows:

$$r_t = \begin{cases} R_{\text{correct}} & \text{if correct identification (legitimate packet)} \\ R_{\text{false\_alarm}} & \text{if false alarm (legitimate packet classified as spoofed)} \\ R_{\text{missed\_detection}} & \text{if missed detection (spoofed packet not detected)} \\ R_{\text{correct\_rejection}} & \text{if correct rejection (spoofed packet identified)} \end{cases}$$

Through repeated interactions and rewards, Bob learns to improve his authentication policy, thus enhancing the network's resilience to spoofing attacks.

Elnaggar and Bezzo (2018) introduce a method to predict and recover from cyber–physical attacks on UAV (Unmanned Aerial Vehicle, aircraft operating without a human pilot on board) using Inverse Reinforcement Learning. The focus is on scenarios where UAVs have to reach a particular position and the attackers try to manipulate the sensor data to disrupt its navigation. In this scenario the admissible **actions** refer to specific movements or adjustments the UAV can make, such as changing direction, speed, or altitude. **States and observations** represent the various conditions or positions of the UAVs within its environment, such as the UAV's geographic coordinates, velocity, altitude, along with sensor readings from gyroscope, accelerometer and GPS.

The **policy** refers to the observed behavior of the UAV, such as its chosen path or maneuvers in response to the environment. IRL uses this observed behavior to infer the underlying **reward function** that the UAV is implicitly trying to maximize. The authors assume that the UAV is following an optimal policy, which reflects its operational goals and constraints. By analyzing the UAV's actions over time, IRL seeks to uncover the reward structure that best explains why the UAV behaves as it does, particularly under adversarial conditions. This inferred reward function is then used to evaluate the success of different actions in maintaining system integrity and achieving the UAV's objectives. The estimated reward function $R(s, a)$ will be based on the system's operational objectives and the influence of potential adversarial interference. It might be formulated as:

$$R(s, a) = -\left(\alpha \cdot d(s, s_{goal}) + \beta \cdot I(s)\right)$$

where $d(s, s_{goal})$ is the distance from the current state s to the goal state s_{goal}, $I(s)$ is an indicator function that penalises unsafe states and α and β are weighting factors. The algorithm uses Bayesian IRL within a Markov Decision Process framework, applying Monte Carlo Markov Chain sampling to predict the attacker's intentions. By exploring different actions, the system refines its predictions about the attacker's goals and identifies which sensors might be compromised.

The system's effectiveness is demonstrated through simulations involving a UAV navigating a stochastic environment.

Xu et al. propose a series of works (Xu, 2010; Xu & Luo, 2007; Xu & Xie, 2005) focusing on the development of temporal-difference-based techniques for anomaly detection in cybersecurity. In their initial work, they introduced TD-SAD (temporal-difference-based sequential anomaly detection) as a method to combat multi-stage cyber attacks in computer systems, showcasing its high detection rates and low false alarm rates across various types of program traces. The strength of TD-SAD lies in its ability to detect subtle, evolving attack patterns over time, particularly in multi-stage attacks where the adversary's behavior may change gradually.

Building upon this foundation, the authors extended TD-SAD to improve anomaly detection capabilities specifically for host-based intrusion detection systems (IDS). This adaptation demonstrated that reinforcement learning techniques could be effectively applied to enhance security in host environments by continuously learning from the system's behavior and adapting to novel threats without requiring extensive retraining.

In their later work, Xu et al. further refined their approach by proposing a novel sequential anomaly detection method using temporal-difference learning principles for host computers. This approach stands out for its ability to model complex sequential behaviors within host systems more efficiently, even in the absence of prior knowledge about the system's processes. By leveraging TD learning, the model can autonomously identify deviations from normal behavior, improving both detection speed and accuracy while maintaining low false alarm rates.

Xiao et al. (2018) explore the security vulnerabilities inherent in Mobile Edge Computing systems. The paper uses RL Frameworks to enhance the security of MEC systems in 5G mobile networks. The study focuses on defending against jamming attacks, denial of service (DoS) attacks and various spoofing attacks, utilizing techniques like Q-learning and Deep Q-Network (DQN). The RL-based approach is evaluated through simulations, revealing its efficacy in improving MEC security while maintaining low overhead. The **state** encompasses variables like received jamming power, radio channel bandwidth, battery levels and user density. **Actions** involve decisions on data offloading, transmit power, channel selection and edge node connectivity. The **reward function** integrates several critical factors:

$$R = w_1 \cdot \text{SINR} - w_2 \cdot \text{BER} - w_3 \cdot E$$

where Signal-to-Interference-plus-Noise Ratio is a measure of signal quality $\text{SINR} = \frac{P_{\text{signal}}}{P_{\text{interference}} + P_{\text{noise}}}$ ratio between power of the received signal and the sum of power of interference from other sources with power of noise. Such expression is used to quantify the quality of a wireless communication signal, considering the impact of interference and noise (higher SINR values indicating better performance). Then BER represents the bit error rate and E denotes energy consumption, with w_1, w_2 and w_3 being weights that balance the importance of each component.

Feng et al. (2020) address the challenge of defending against application-layer distributed denial-of-service (L7 DDoS) attacks, which exploit application-layer requests that appear legitimate to overwhelm server functions. Traditional DDoS defences struggle with L7 DDoS attacks due to their subtle nature at the transport and network layers, that is where defense mechanisms usually intervene. The authors propose a defence mechanism using RL, where an agent learns to mitigate these attacks through a *multi-objective reward function*. This function balances the aggressive mitigation of malicious requests during severe attacks with conservative mitigation to minimize collateral damage to legitimate traffic under normal conditions. Their evaluation demonstrates that the proposed approach effectively mitigates 98.73% of malicious events.

Oh and Iyengar (2019) introduce a novel framework for detecting anomalies in sequential data using IRL. The primary objective is to infer the underlying reward function of a decision-making agent from its behavior data, which in turn helps identify deviations from normal behavior that may indicate anomalies. The proposed method uses a neural network to model the reward function and incorporates Bayesian approaches to account for model uncertainty, thus enhancing the reliability of the anomaly detection. The reward function is estimated using trust-region policy optimization (TRPO), a state-of-the-art policy gradient method. The framework generates new trajectories and updates the reward parameters iteratively. The reward function $r_\theta(s, a)$ is modeled using a neural network, where θ represents the parameters of the network. The objective is to optimize θ such that the behavior of the agent, represented by its state–action pairs (s, a), is accurately captured. The reward function is updated iteratively using the gradient descent method

The total reward for trajectory τ_j, $R(\tau_j|\theta)$, is given by:

$$R(\tau_j|\theta) = \sum_{(s,a)\in\tau_j} r_\theta(s, a). \tag{1}$$

Additionally, the normality score $n(s, a)$ for a new observation (s, a) is defined as:

$$n(s, a) = \frac{r_\theta(s, a) - \bar{r}}{\sigma_r}, \tag{2}$$

Table 3
Summary of Single-agent RL for Host-based security.

Ref.	Algorithm	Threats	Evaluation metrics	Reward elements
Liu et al. (2017)	Q-Learning	Spoofing	Authentication accuracy, false alarms, missed detections	True/false Positives/negatives
Elnaggar and Bezzo (2018)	IRL	Sensor data poisoning	System integrity, goal achievement	Distance to goal, unsafe states
Xu and Xie (2005)	SARSA	Multi-stage cyber attacks (e.g., APTs)	Detection rates, false alarm rates	Anomaly probabilities
Xu and Luo (2007)	Q-Learning	IDS	Anomaly detection accuracy	Anomaly probabilities
Xiao et al. (2018)	Q-Learning and DQNs	Spoofing, authentication and secure mobile offloading for MEC security	Secure mobile offloading, authentication, caching efficiency	Secure offloading, authentication success, caching efficiency
Xu (2010)	SARSA	Anomaly detection	Anomaly detection accuracy	Anomaly probabilities
Feng et al. (2020)	Multi-objective RL	Application-layer DDoS attacks	Malicious event mitigation	Aggressive mitigation, conservative mitigation
Oh and Iyengar (2019)	IRL	Anomaly detection	Anomaly detection reliability	Normality score

where $\bar{r}$ is the average reward over all observations and σ_r is the standard deviation of the reward. The normality score helps in identifying anomalies: an observation is considered anomalous if its normality score falls below a predefined threshold ϵ. This threshold can be dynamically adjusted based on the specific requirements of the application, ensuring a balanced detection of anomalies across various behavioral patterns.

Discussion. In conclusion, the single-agent RL Frameworks for host-based security summarized in Table 3 demonstrate significant potential in addressing various cybersecurity challenges. These studies illustrate the versatility of RL algorithms in enhancing authentication accuracy, predicting and recovering from cyber–physical attacks and improving anomaly detection in host systems. A notable observation is the frequent use of Temporal Difference learning across multiple studies, adopted due to its simplicity and effectiveness in sequential decision-making tasks. Despite the potential advantages of deep learning, such as better handling of high-dimensional data and complex patterns, its application in single-agent RL for host-based security remains scarce. Furthermore, a significant number of studies focus on anomaly detection, particularly in host-based IDS. The anomalies detected typically relate to deviations from normal behavior patterns in network traffic, system logs, or user activities. This emphasis on anomaly detection underscores the critical role of identifying and responding to unusual activities to maintain robust security. Consequently, the frequent use of anomaly probabilities as reward elements highlights their importance in maintaining robust security.

In most of these studies, tabular algorithms like Q-learning and SARSA are commonly employed due to the relatively low-dimensional state spaces of host environments. For more complex environments, deep reinforcement learning algorithms, such as DQN, have been adopted to handle higher-dimensional state representations. These algorithms are particularly effective for detecting and mitigating malware, as demonstrated in several experimental setups.

5.2. Single-agent RL in SDN-based security

The integration of SDN with RL has opened new avenues for enhancing network security. This synergy is particularly effective in mitigating various cyber threats. In this section, we explore several state-of-the-art approaches using single-agent RL to secure SDN environments (see Table 4 at the end of this subsection).

Liu et al. (2018) present a DRL approach for mitigating Distributed Denial of Service (DDoS) attacks in SDNs. The system employs a Deep Deterministic Policy Gradient algorithm, using the OpenFlow protocol, that is a communication protocol that gives access to the forwarding plane of a network switch or router over the network. It enables a network controller to determine the path of network packets across a network of switches, routers and access points for network visibility. The **state** space captures network features from OpenFlow switches; the **action** space configures bandwidth limits for hosts using OpenFlow meters; the **observation** process continuously monitors network traffic. The **reward function** is defined as:

$$\text{reward} = \begin{cases} -1 & \text{if Load}_s > U_s \\ \lambda p_b + (1-\lambda)(1-p_a) & \text{if Load}_s \leq U_s \end{cases}$$

$Load_s$ represents the server load, U_s is the upper load boundary, p_b is the percentage of benign traffic reaching the victim server, p_a is the percentage of attack traffic reaching the server and λ is a weighting parameter. This reward function ensures that the agent is penalized for server overload and rewarded for maximizing the delivery of benign traffic while minimizing attack traffic. In summary, the DRL-based approach dynamically adjusts bandwidth allocations to effectively mitigate DDoS attacks.

Guo et al. (2020) explore the use of DRL to enhance secure routing in SDN for IoT environments. The proposed Deep Q-learning Secure Routing Protocol (DQSP) utilizes a Convolutional Neural Network (CNN) and a DDPG agent to optimize routing decisions, aiming to improve network performance under Gray Hole (Tripathi, Gaur, & Laxmi, 2013) and DDoS attacks. Grey Hole attack is a common type of attack in Wireless Sensor Network (WSN) where malicious nodes may constantly or randomly drop packets and therefore reduce the efficiency of the networking system. The study evaluates the approach through simulations, measuring packet delivery ratio, end-to-end delay and the probability of routing through attacked nodes. The **reward function** considers multiple factors including packet loss rate, forwarding delay, flow table status, propagation delay and link packet loss rate. It is a weighted sum of attack-related rewards and QoS-related reward.

Phan et al. (2019) present a novel approach to detect and mitigate stealthy DoS attacks (characterized by low-rate or slow attack patterns that are designed to evade detection mechanisms) in SDN-based networks using a Q-Learning-based RL framework. The framework, named Q-MIND, exploits a global view of the network state provided by SDN to analyze traffic patterns and detect anomalies indicative of stealthy DoS attacks. The principal features in the Q-MIND framework involve the careful selection and analysis of network traffic metrics. These include parameters such as the average number of packets per flow and the average packet size within a flow. The system then uses these metrics to train a Q-Learning agent, which learns to recognize patterns indicative of stealthy DoS attacks and deploys an optimal policy to detect and mitigate these threats effectively. The **actions** are a set of algorithms that will be applied to analyze the network traffic and identify potential attacks The **State** is defined by evaluation metrics such as precision,

recall, F-score, accuracy and false alarm rate. The **Observation** consists of traffic flow statistics and detection performance. The **reward** function combines multiple evaluation criteria like precision, recall and false alarm rate and is defined as follows:

$$r(s,a) = W_{Pr}Pr + W_{Re}Re + W_{Fs}Fs + W_{Ac}Ac + W_{Fa}e^{-Fa}$$

where Pr is precision, Re is recall, Fs is F-score, Ac is accuracy, Fa is false alarm rate and $W_{Pr}, W_{Re}, W_{Fs}, W_{Ac}, W_{Fa}$ are weight factors related to the corresponding evaluation criteria. The paper acknowledges that existing methods do not fully address the detection problem, suggesting that optimizing the feature set and machine learning algorithms could further improve performance.

Akbari et al. (2020) introduce a framework for using RL to detect and mitigate Advanced Persistent Threats (APTs) in SDN environments. The authors implement a Neural Fitted Q-learning agent to handle the sequential decision-making required for effective threat mitigation. The framework operates within a simulated network environment, using Mininet to emulate benign and malicious host behavior. The SDN infrastructure provides a global view of the network, which the RL agent uses to enforce security policies dynamically. The **reward function** is designed to balance the quality of service for benign hosts against the effectiveness of attack mitigation, calculated as:

$$R = \text{QoS}_{\text{benign}} - \text{Success}_{\text{attack}}$$

where $\text{QoS}_{\text{benign}}$ represents the quality of service metrics from benign hosts and $\text{Success}_{\text{attack}}$ represents the success rate of attacks by malicious hosts. The goal is to maximize $\text{QoS}_{\text{benign}}$ while minimizing $\text{Success}_{\text{attack}}$. The **action** space is simplified by adopting virtual networks with varying security levels, allowing the RL agent to dynamically adjust security measures based on observed threats. The paper highlights the need for improved methods to handle unseen attacks, reduce reward signal variance and refine network observation techniques.

Phan et al. (2020) presents an approach to detect and mitigate DDoS attacks in SDN using a combination of DRL and Support Vector Machine. The primary focus is on volumetric (flooding with a high volume of traffic) and stealthy DDoS attacks. The proposed system, DEEPGUARD, operates in a simulated network environment using the *MaxiNet* framework. Key components include a traffic flow matching control mechanism and an anomaly detection system. The traffic flow matching control utilizes a Double Deep Q-Network to dynamically adjust the flow matching schemes, aiming to maximize traffic granularity while preventing switch performance degradation. The reward function for the DDQN algorithm is defined as:

$$R_i(s,a) = \begin{cases} \frac{1}{f_i}\sum_{x=1}^{f_i}\Theta_x & \text{if } 0 < f_i < f_{\text{cap}} \\ 0 & \text{if } f_i = f_{\text{cap}} \end{cases}$$

where f_i is the current total number of flow entries, Θ_x represents the number of enabled match fields in flow entry x and f_{cap} is the capacity of the SDN switch.

Zolotukhin et al. (2020) utilize deep Q-network and proximal policy optimization algorithms to train an intelligent defense system capable of responding to SSH password brute-force attacks, DNS tunneling and Slowloris DDoS attacks, designed to consume minimal bandwidth while causing significant disruption and used specifically to attack web servers opening a large number of connections. The experimental setup involves a small network environment implemented using the open-source cloud computing platform OpenStack integrated with the SDN controller Opendaylight. The implementation is later switched to Docker containers for better resource efficiency since all appliances used in the experiments had the same x86_64 architecture. The environment involves creating virtual copies of the network, running the same applications as IoT devices in the target network and using Openflow-enabled OpenVSwitches for routing traffic. The RL agent interacts with the SDN controller to enforce security policies, taking actions such as redirecting traffic to security appliances, dropping suspicious connections, or allowing previously blocked traffic. Key features of the system include the use of cloud computing and network virtualization to create virtual security appliances, such as intrusion detection systems, honeypots and firewalls. The **reward function** is designed to incentivize the RL agent to maximize network security while minimizing the impact on legitimate traffic. It is mathematically represented as:

$$R(a) = \sum_i \left(\alpha \cdot \text{legit\_packets}_i - \beta \cdot \text{malicious\_packets}_i\right)$$

where α and β are positive coefficients for legitimate and malicious packets, respectively. The study concludes that while RL-based defense mechanisms show promise in reducing the impact of network attacks in small environments, further work is needed to address scalability and reward function specification challenges.

Kim et al. (2021) address the detection of network attacks such as Advanced Persistent Threats (APTs) and TCP SYN flooding attacks using a novel technique involving Deep Deterministic Policy Gradient in software-defined networks. APTs are sophisticated, long-term cyberattacks that target specific organizations to steal data or cause damage, while TCP SYN flooding attacks are a type of denial-of-service (DoS) attack that overwhelms a network by exploiting the TCP three way handshake process. The proposed approach aims to optimize traffic sampling by dynamically adjusting the sampling rates at various switches to improve monitoring efficiency, load balancing and reducing steering overhead.

The **reward function** $R(s_t, a_t)$ is defined as:

$$R(s_t, a_t) = (r_f \cdot r_u) - r_v$$

where r_f is the flow fair-share reward, r_u is the load-balancing reward and r_v is the steering overhead penalty. The flow fair-share reward r_f is designed to ensure an equitable allocation of network resources among all network flows. This reward incentivizes the agent to distribute traffic sampling resources uniformly, preventing any single flow from dominating bandwidth and ensuring balanced network performance. The approach faces challenges such as handling dynamic network conditions and the complexity of implementing DRL in real-time. Future improvements could focus on better managing large-scale network environments.

Zhang, Qiu et al. (2023) present a novel approach to enhance the security and efficiency of SDN against DDoS attacks. The proposed method, Trust-Based Proximal Policy Optimization (TBPPO), integrates a trust value mechanism based on Kullback–Leibler divergence and a node diversity mechanism, which increases security by diversifying the routing paths to make it more challenging for attackers to compromise multiple paths. TBPPO is an improved PPO algorithm, considering security, network delay and variations in multi-path delays. The algorithm uses an enhanced Depth-First Search to pre-compute path sets and optimize multi-path routing. The principal features of the algorithm include mitigating network fluctuations, enhancing robustness and addressing congestion issues. The **reward function** used in the TBPPO algorithm is formulated as:

$$R(\tau) = \sum_{o \in O} \alpha_o r_o(\tau)$$

where α_o represents the weight of reward type o and $r_o(\tau)$ represents the reward value of type o during the time window τ. The reward types include delay, delay variation, KL trust value, node diversity and node redundancy. Improvements can be made to enhance computational efficiency and real-time application suitability.

Discussion. In summary, the integration of RL with SDN offers a powerful combination for enhancing network security. The studies reviewed (see Table 4) demonstrate how single-agent RL can effectively mitigate various cyber threats in SDN environments, improving network performance and security. A common trend observed is the frequent use of Deep RL Frameworks such as Deep Deterministic Policy Gradient

Table 4
Summary of Single-agent RL for Network-based security with SDN.

Ref.	Algorithm	Threats	Evaluation metrics	Reward elements
Liu et al. (2018)	DDPG	DDoS	Server load, traffic	Server load, benign traffic, attack traffic
Guo et al. (2020)	DQN	Gray Hole, DDoS	Packet delivery ratio, end-to-end delay	Packet loss rate, forwarding delay, flow table status, propagation delay, link packet loss rate
Phan et al. (2019)	Q-Learning	Stealthy DoS	Precision, recall, F-score, false alarm rate	Precision, recall, F-score, false alarm rate
Akbari et al. (2020)	DQN	APTs	QoS, attack mitigation	QoS, attack mitigation effectiveness
Phan et al. (2020)	Double DQN	DDoS	Flow entry efficiency, switch performance	Number of enabled match fields, flow entry capacity
Zolotukhin et al. (2020)	DQN, PPO	SSH brute-force, DNS tunneling	Network security	Legitimate packets, malicious packets
Kim et al. (2021)	DDPG	APTs, TCP SYN flooding	Monitoring efficiency, load balancing	Flow fair-share reward, load-balancing reward, steering overhead penalty
Zhang, Qiu et al. (2023)	PPO	DDoS	Network delay, delay variation, trust value	Delay variation, node diversity, node redundancy

and Deep Q-learning, which leverage SDN's global network view for precise traffic management. These techniques leverage the centralized control of SDN to optimize traffic routing and detect anomalies. Specifically, DQN has been favored for its ability to handle discrete action spaces, whereas PPO is employed for tasks requiring continuous control or policy optimization. These choices are influenced by the need to balance scalability and response time in SDN contexts. Additionally, many studies focus on addressing DDoS attacks, highlighting the critical importance of defending against these prevalent threats. Common evaluation metrics include server load, benign traffic, attack traffic and various quality of service (QoS) indicators, which are also commonly used as reward elements to guide the RL agents.

5.3. Multi-agent RL in SDN-based security

This category focuses on methods that apply multi-agent RL Frameworks within SDN-based security frameworks. The integration of multi-agent RL in SDN environments offers a powerful approach to managing complex and dynamic networks. SDN's centralized control plane inherently facilitates the collection and dissemination of network-wide information, enabling various configurations of multi-agent RL—whether centralized, decentralized, or CTDE. In all these configurations, each agent utilizes the centralized SDN controller to access critical network data and execute actions. This setup allows for a more flexible and adaptive security posture, where agents can respond to evolving threats in a coordinated manner, thereby enhancing the overall protection of the network (see Table 5 at the end of this subsection).

Janakiraman and Deva Priya (2023) propose a DRL approach exploiting LSTM networks for mitigating DDoS attacks in fog-assisted cloud environments. The technique involves multiple agents collaborating in a centralized network-based approach to identify and mitigate DDoS attacks at the network layer. The DRL algorithm used follows Q-learning or Policy Gradient methods. The agents learn to differentiate between legitimate and malicious packets, employing SDN controllers for network traffic analysis and filtering. Additionally, LSTM networks are incorporated to handle the temporal nature of network traffic data, enabling agents to capture long-term dependencies and effectively classify incoming packets as either legitimate or malicious. In this context, the reward is defined as the successful identification and mitigation of DDoS attacks while minimizing false positives and maintaining the availability of legitimate network services. The **reward function** $R(s,a)$ can be mathematically represented as:

$$R(s,a) = \begin{cases} 1 & \text{if the packet is correctly identified and mitigated} \\ -1 & \text{if the packet is incorrectly identified} \\ 0 & \text{otherwise} \end{cases}$$

The proposed DRL-LSTM model is implemented using Mininet emulator for constructing the network topology and FloodLight controller for SDN. The model is trained and tested using the Hogzilladataset,[1] derived from CTU-13Botnet[2] and ISCX2012[3] intrusion detection system datasets. The experimental results show high accuracy in identifying and mitigating DDoS attacks, with a training accuracy of 99.59% and testing accuracy of 98.98%.

Yungaicela-Naula et al. (2022) describe a multi-agent approach where each agent is responsible for managing a specific bidirectional connection in the network. The agents operate in a decentralized manner, each learning and executing its own mitigation strategy independently. This approach helps in parallel mitigation of DDoS attacks. The focus is on slow HTTP read attacks, simulated within a real network using Mininet. The DRL-based IPS utilizes a **reward function** defined as follows:

$$r_k = \begin{cases} 1 & \text{(Terminal state), if } d_k = 0 \text{ and } b_k > 0 \\ 0 & \text{if } d_k = 0 \text{ and } b_k = 0 \text{ and } a_2 \\ 0 < \psi < 1 & \text{if } d_k = 0 \text{ and } b_k = 0 \text{ and } a_3 \\ -e_{mk} & \text{otherwise.} \end{cases}$$

Here, d_k indicates the attack state, b_k the presence of traffic and a_2 and a_3 represent actions. While the framework effectively blocks malicious connections, it does not manage opened service threads on the victim server, leading to prolonged service unavailability for legitimate users. Future work includes integrating application-layer capabilities, which involve managing application-layer processes, to close service threads that remain open after an attack. Additionally, the authors suggest optimizing hyperparameters and enhancing the RL algorithm for better performance and scalability.

Rezapour and Tzeng (2022) present a method to counteract Link-Flooding Attacks (LFAs) using a combination of SDN and DRL. The approach focuses on real network applications and utilizes a decentralized multi-agent system. The evaluation of the proposed system is conducted using Mininet with Ryu as an OpenFlow controller for

[1] https://ids-hogzilla.org/dataset/.
[2] https://www.stratosphereips.org/datasets-ctu13.
[3] https://www.unb.ca/cic/datasets/ids.html.

managing the network and adopt Iperf to generate flow traffic. The **State** space S includes link congestion probabilities, path indicators, link status and a shared state-value function. The **Action** space A involves selecting next-hop neighbors for data forwarding. The **Reward function** is designed to penalize invalid actions and provide varying rewards based on the agent's success in reaching the target node. The reward function is given by:

$$r = \begin{cases} -0.02 & \text{if } a \notin A(s) \\ 0 & \text{if } a \in A(s) \text{ and } CF(conditionflag) = 0 \\ -0.001 & \text{if } a \in A(s) \text{ and } CF = 1 \text{ and } des(a) \neq n_{trg} \\ -1 & \text{if } a \in A(s) \text{ and } CF = 1 \text{ and } des(a) = n_{trg} \text{ and } \\ & \quad p_{src,trg} = p'_{src,trg} \\ r_0 & \text{if } a \in A(s) \text{ and } CF = 1 \text{ and } des(a) = n_{trg} \text{ and } \\ & \quad p_{src,trg} \neq p'_{src,trg} \end{cases}$$

where $CF(conditionflag)$ is a condition flag that indicates whether the path construction is complete. It takes the value of 0 if Path construction is not yet complete or 1 indicating Path construction is complete. The set $A(s)$ denotes the available actions in state s. If $a \notin A(s)$, it implies that the action a is invalid for the given state and the agent receives a penalty. Instead, $des(a)$ represents the destination node associated with the action a. This is the node to which the agent directs traffic as part of the decision-making process. The target node n_{trg} is the intended final destination in the network that the agent aims to reach or protect. Term $p_{src,trg}$ denotes the newly constructed path from the source node to the target node after the agent has taken the action a. Term $p'_{src,trg}$ represents the existing path between the source node and the target node before the agent takes action a. Finally, term r_0 is a predefined positive reward that the agent receives when it successfully identifies a new, non-flooded path from the source to the target node. This reward encourages the agent to explore alternative paths to avoid potential attacks. The algorithm operates in a real network environment, addressing practical challenges associated with LFAs. While effective, the authors note that further enhancements can be made to improve scalability and handle more dynamic network conditions.

Simpson et al. (2020) introduce a RL approach for mitigating DDoS attacks in a SDN environment using mininet and RYU controller. The RL Frameworks employed is semi-gradient Sarsa, chosen for its lower latency and simpler decision boundaries. The principal features utilized by the model include global state (network link load observations), source IP address, last action taken, flow duration and size, correspondence ratio, Δ send/receive rate, mean inter-arrival time (IAT), per-window packet count and mean packet size per window. Despite these features, challenges remain in protecting legitimate UDP traffic and designing effective reward functions without relying on heuristic estimates. The **reward function**, based on the volume of legitimate traffic received and network load, is defined as:

$$R_{s,t} = (1 - c_{s,t}) \frac{g(\text{load}_t(s))}{\text{traffic}_s} - c_{s,t} \tag{3}$$

where $c_{s,t} = [\max(\text{load}_{\uparrow t}(s), \text{load}_{\downarrow t}(s)) > U_s]$. The **state** represents network conditions and flow statistics. The **actions** are the decisions to filter or allow traffic based on current state. The **observations** are traffic patterns and network load. The system operates as a decentralized multi-agent system, where agents share experiences but not weight vector updates. The implementation has a repository available at https://github.com/FelixMcFelix/rln-dc-ddos-paper.

Dake et al. (2021) propose a novel approach using Multi-Agent Deep Deterministic Policy Gradient (MADDPG) to address DDoS attacks in a SDN enabled IoT environment. The framework follows the principles of CTDE MARL where agents perform decentralized actions but are trained in a centralized manner. The setup was simulated using Mininet. The proposed MADDPG framework comprises two intelligent agents in the controller and an observed environment consisting of data forwarding devices and IoT devices. Agent 1 is responsible for genuine traffic burst, ensuring that legitimate spikes in traffic do not overwhelm the network infrastructure, and elephant flow multipath routing (Zaw & Maw, 2019) to prevent congestion and improve overall network performance, while agent 2 is responsible for DDoS detection and blocking. The **state** space $s(t)$ is defined as a traffic matrix TM, representing the current network conditions, the **action** space includes actions for routing and dropping packets, the **observation** is network metrics and conditions observed by agents. The **reward function**, with the goal to minimize DDoS impacts and optimize network performance, is as follows:

$$R_{\text{agent1}}(t) = \frac{1}{U}, R_{\text{agent2}}(t) = \begin{cases} 1, & \text{if flow\_freq}_i > RR \\ -1, & \text{if flow\_freq}_i \approx \text{loss} \end{cases}$$

Future research directions include extending the threat detection capabilities and deploying the framework in distributed controller environments.

Discussion. In summary, the application of multi-agent RL Frameworks in SDN environments (see Table 5) provides a robust and scalable approach to network security. A consistent pattern across studies is the reliance on advanced deep RL methods, such as Deep Q-learning and Policy Gradient approaches, which harness the collaborative potential of multiple agents to effectively handle sophisticated cyber threats. The emphasis on DDoS attack mitigation underscores the pressing need to safeguard networks against these widespread dangers. Key evaluation metrics often include factors like mitigation rate, false positive rate, service availability and network load, which are also employed as reward elements to optimize RL agent performance. By capitalizing on the inherent strengths of SDN and multi-agent frameworks, these methods considerably bolster the security and resilience of network infrastructures.

In multi-agent RL applications for SDN-based security, algorithms like Multi-Agent Deep Deterministic Policy Gradient (MADDPG) and Independent Q-Learning (IQL) are frequently used. These methods are capable of coordinating multiple agents in tasks like intrusion detection and resource allocation. MADDPG, in particular, enables agents to learn both cooperative and competitive strategies, making it a popular choice for defending against distributed attacks in SDN environments.

5.4. Multi-agent RL in network-based security

This category involves approaches where multiple learning agents work collaboratively to secure the network (see Table 6 at the end of this subsection). Each agent is responsible for a segment of the network, making decisions while communicating with other agents.

Veluchamy and Kathavarayan (2022) propose a novel system called *DARLH* (Deep Adaptive Reinforcement Learning for Honeypots) to detect and mitigate DDoS attacks. The DARLH system uses Deep Learning techniques, specifically combining Deep Recurrent Neural Networks and Deep Adaptive Reinforcement Learning based Intrusion Detection System (IDS) agents. The **state** represent the current configuration or context of the honeypot environment, which provides valuable insights into attack techniques by misleading attackers into interacting with fake targets, thus enhancing overall network security, including network traffic patterns and system resource utilization. The **actions** are defensive maneuvers the system can undertake to respond to perceived threats. The **reward function** incentivizes successful threat detection and mitigation actions.

The DARLH system operates on both *single-agent* and *multi-agent*. At the single-agent level, a single RL agent autonomously learns and makes decisions based on its observations of network traffic. At the multi-agent level, multiple agents collaborate, sharing information and coordinating actions to enhance the overall security posture of the system. This dual-level architecture combines the adaptability and autonomy of single-agent systems with the collaborative and coordinated

Table 5
Summary of Multi-agent RL for Network-based security with SDN.

Ref.	Algorithm	Threats	Key metrics	Reward elements
Janakiraman and Deva Priya (2023)	Deep Q-learning or Policy Gradient with LSTM	DDoS	Mitigation rate, false positives, service availability	Successful attack mitigation
Yungaicela-Naula et al. (2022)	Deep Q-Learning	Slow HTTP read attacks	Blocking rate, service availability	Attack state, network load
Simpson et al. (2020)	Semi-gradient Sarsa	DDoS	Legitimate traffic volume, network load	Legitimate traffic volume, network load
Rezapour and Tzeng (2022)	Variant of Monte Carlo algorithms	Link-Flooding Attacks	Link congestion probabilities, path indicators	Condition flag: 0 path is being constructed, 1 path construction complete
Dake et al. (2021)	Multi-Agent Deep Deterministic Policy Gradient	DDoS	Impact minimization, network performance	Flow frequency, routing decisions, loss of traffic

capabilities of multi-agent systems, offering a holistic approach to network security. The proposed system is implemented with 10 server units and a generic database. Each server in honeypot controls 25 client machines. The proposed DARL and DRNN IDS agents are implemented in servers and clients. For detecting DoS attacks, the IDS agents run rule based classification procedures and signature mapping techniques. The implementation uses Weka 3.0 and Python 3.7, evaluating the system using metrics such as True Positive Rate, False Positive Rate, Classification Accuracy and Error Rate against both internal and external DoS attacks. Future improvements suggest addressing multiple attack detection strategies to further enhance the system's robustness.

Dasgupta et al. (2022) present a DRL approach to detect and mitigate GPS spoofing attacks in autonomous vehicles. The study employs a centralized multi-agent system using low-cost in-vehicle sensor data, such as GPS and CAN, within a real-world vehicular network. The **state** represents the current configuration of the autonomous vehicle, including data from sensors like GPS coordinates, speed, steering angle and accelerator position. **Actions** involve decisions made by the system to detect and mitigate spoofing attacks, such as adjusting thresholds for GPS data validation. **Observations** consist of real-time data collected from in-vehicle sensors. The **reward function** is designed to prioritize accurate detection of spoofing attacks:

$$R = \begin{cases} +1 & \text{if the agent's detection matches the ground truth} \\ -100 & \text{if the agent's detection does not match} \\ & \text{the ground truth} \end{cases}$$

The deep RL model employs a Deep Q Network to approximate the optimal policy. The deep RL model achieves high accuracy, recall and precision in detecting turn-by-turn GPS spoofing attacks, with performance metrics ranging from 99.99% to 100%. The study highlights the importance of collaboration among agents in a centralized multi-agent framework to enhance overall system performance. Future work involves improving the model's robustness to various driving scenarios and addressing the security of in-vehicle sensors and data privacy.

Malialis and Kudenko (2015) introduce a novel approach called Coordinated Team Learning (CTL) to mitigate DDoS attacks using MARL. The CTL approach combines hierarchical team-based communication, task decomposition and team rewards to improve scalability and performance in network intrusion response. The study focuses on distributed rate-limiting mechanisms, particularly for DDoS attacks, and demonstrates the effectiveness of the CTL approach through experiments involving up to 100 reinforcement learning agents. The approach is tested using a network emulator to simulate realistic network conditions. These agents use RL to dynamically adjust router throttling mechanisms, effectively mitigating DDoS attacks impact on network performance and availability. The approach is *decentralized*, as each agent operates independently, making decisions based on local observations and interactions with the environment. However, they collaborate indirectly by collectively improving the overall network resilience through their individual actions.

Bhagyashree Deokar (2012) propose a cooperative learning method for IDS based on multi-agent systems. The system architecture involves multiple agents distributed across different hosts, each responsible for monitoring network connections and system log files. These agents collaborate in a *decentralized* manner by sharing information and making local decisions based on their observations, contributing to a collective decision about whether an intrusion has occurred. The decision-making process utilizes influence diagrams and Bayesian networks to model uncertainty and optimize decision outcomes.

Bhosale et al. (2014) propose an approach to IDS using a multi-agent framework and MARL. It addresses the limitations of traditional single-agent IDS, which struggle to handle the complexity and real-time demands of modern network security. By employing a multi-agent system, each agent possesses partial information and collaborates with others to improve decision-making capabilities. The decision-making process is facilitated by influence diagrams, which represent probabilistic relationships between events and guide local decision-making. This approach leans towards *decentralization*, as agents collaborate but maintain their autonomy in decision-making.

Shamshirband et al. (2014) introduce a novel approach called Cooperative Game-based Fuzzy Q-learning (G-FQL) for detecting and mitigating DDoS attacks, specifically flooding attacks, in Wireless Sensor Networks (WSNs). The technique integrates game theory with fuzzy Q-learning, creating a three-player game involving the sink node, base station and an attacker. The system is designed to operate within a simulated environment using the NS-2simulator,[4] where it is evaluated on metrics such as detection accuracy, counter-defense, network lifetime and energy consumption. The G-FQL model features a Centralized Multiagent System, where agents (sink nodes and base station) collaborate to defend against attacks. The fuzzy Q-learning algorithm enables the agents to adapt and learn optimal strategies based on environmental observations. The model simplifies the scenario by focusing on DDoS flooding attacks and assumes an alarm event threshold to trigger defense actions. The **actions** refer to the decisions made by the sink node and base station, such as increasing monitoring, isolating suspected nodes, or triggering counter-defense mechanisms to mitigate a detected attack. The **state** represents the current condition of the network, including factors like node energy levels, traffic patterns and detection of abnormal behavior indicative of a potential attack. **Observations** are the network parameters and events that agents (sink nodes and base stations) perceive, including traffic anomalies, energy consumption rates and node behaviors that inform the decision-making process. The **reward function** is structured to incentivize successful detection and defense while penalizing failures. The general form of the reward can be expressed as:

$$R(s, a) = \text{Detection Reward} - \text{Penalty for Failure}$$

where s represents the state and a represents the action taken by the agent. While the G-FQL model shows promise in improving detection

[4] https://www.isi.edu/websites/nsnam/ns/.

Table 6
Summary of Multi-agent RL for Network-based security.

Ref.	Algorithm	Threats	Key metrics	Reward elements
Veluchamy and Kathavarayan (2022)	Deep RL	DDos	Security posture, attack analysis	Detection effectiveness, attack mitigation
Dasgupta et al. (2022)	DQN	GPS spoofing	Detection accuracy, system integrity	Detection success
Shamshirband et al. (2014)	Cooperative Fuzzy Q-learning	DDoS	Detection accuracy, network resilience	Attack detection rate, Power cost for defense against attacks, % of attacks detected
Malialis and Kudenko (2015)	Sarsa	DDoS	Network performance, service status	Server Load, Legitimate Traffic
Bhagyashree Deokar (2012)	Q-learning	Intrusions	Detection accuracy, collective decision-making	Types and structure of log entries, knowledge bases of normal behavior
Bhosale et al. (2014)	Q-Learning	Intrusions	Decision accuracy, collaboration efficiency	Confusion matrix
Sethi et al. (2021)	Deep Q-Learning	Intrusions	Detection accuracy rate, robustness	Accuracy, precision, recall, F1-Score

accuracy and energy efficiency, future improvements could involve real-world testing, expanding attack scenarios and refining the reward structure for better adaptability in diverse environments.

Sethi et al. (2021) explore a novel approach to intrusion detection adopting DRL agents. The agents, deployed in routers, predict Q-values for actions based on network packet data. A central IDS aggregates these Q-values using an attention mechanism to determine the malicious nature of the packets. The **action** A_t taken by an agent is to predict whether a packet is an intrusion (1) or non-intrusion (0). The **state** S_t represents the true underlying condition of the network traffic, indicating whether the traffic is normal or under attack. The **observation** O_t is the feature vector derived from the network packets at time t, including attributes like packet size, duration, protocol type, etc.. The **reward function** provides feedback to the agents based on their prediction accuracy:

$$R = \begin{cases} +1 & \text{if correct intrusion detection} \\ -1 & \text{if incorrect intrusion detection} \\ +0.1 & \text{if correct non-intrusion detection} \\ -0.1 & \text{if incorrect non-intrusion detection} \end{cases} \tag{4}$$

The proposed IDS model combines decentralized multi-agent reinforcement learning with a centralized attention mechanism, enhancing intrusion detection accuracy and robustness. Future work aims to address the challenges posed by sophisticated adversarial attacks and optimize the model for real-time applications.

Discussion. In conclusion, multi-agent RL Frameworks for network-based security (see Table 6) highlight the importance of collaborative intelligence in combating cyber threats over a network. A common trend is the emphasis on various forms of cooperation and communication between agents to enhance detection accuracy and overall network resilience. Many studies focus on mitigating DDoS attacks, which remain a critical threat to network security. Common evaluation metrics include detection accuracy, network performance and collaboration efficiency, reflecting the multifaceted nature of modern cybersecurity challenges. By adopting collaborative capabilities of multiple agents, these approaches dynamically adjust security measures to maintain robust defenses against evolving cyber threats.

For network-based security, multi-agent frameworks typically employ algorithms such as Cooperative Multi-Agent Q-Learning (CMAQ) and policy-gradient methods like PPO. These algorithms excel at optimizing distributed defense strategies across network nodes. For example, CMAQ has been applied to enhance intrusion detection systems by enabling collaboration among multiple agents deployed at various network points.

5.5. Adversarial RL for network and SDN-based security

This category focus on a specific RL Frameworks (see Table 7 at the end of this subsection), termed *adversarial*, where RL agents learn a policy that is actively disrupted by hostile agents (that usually are themselves learning, too). These agents are trained to anticipate and counteract sophisticated attacks, thereby enhancing the network's resilience against adversarial threats.

Chowdhary et al. (2021) model attacker and defender interactions as a zero-sum dynamic game in an SDN-managed cloud environment, utilizing Deep-Q Learning for policy optimization. The research aims to increase the complexity of attacks by dynamically shifting network configurations, thereby providing a defense mechanism against adaptive adversaries. The **actions** include reconnaissance and exploitation for attackers and no action, shuffle and IP mutation for defenders. The **state** of game is defined in terms of privilege of each player. The state transition depends on the actions of both the players. The transition probability will depend on the complexity of geting access to the vulnerability. The **observation** consists of the network state and attack vectors. Policies are optimized using DQL in a competitive multi-agent setting. The **reward functions** incorporate CVSS scores[5] (Common Vulnerability Scoring System) – that are a standardized way to assess the severity of security vulnerabilities in software and systems scores – attack costs C_A and defense costs C_D:

$$R_{\text{attacker}} = 3 - f(C_A, C_D), R_{\text{defender}} = f(C_A, C_D) - 3$$

The experiments are conducted in a simulated network environment using OpenAI Gym. The results demonstrate that the defender's utility is higher when employing MARL strategies compared to random strategies. The study highlights the need for continuous model evaluation and addresses the challenges posed by the non-stationary nature of the environment.

Novaes et al. (2021) address the use of DRL to detect and defend against Distributed Denial of Service (DDoS) attacks in a SDN context. The study is conducted in a simulated network environment, utilizing adversarial learning techniques to enhance the robustness of the model. The **state** (current network status) includes features like packet flow statistics and network topology. The **action**s taken by the agent involve flagging suspicious traffic, re-routing packets, or deploying mitigations. As its **observation** the agent has access to network traffic patterns, changes in flow metrics and potential anomalies. The policy is trained to minimize the damage caused by the DDoS attack while maximizing detection efficiency. The **reward function** can be represented as

[5] https://nvd.nist.gov/vuln-metrics/cvss.

follows:

$$R_t = -\alpha \cdot \text{Cost}_{attack} + \beta \cdot \text{Detection Accuracy}$$

where α and β are constants representing the balance between minimizing attack impact and maximizing detection accuracy.

Turner et al. (2022) employ a Breach and Attack Simulation (BAS) framework to simulate the interactions between attackers and defenders, modeled as a zero-sum game. The primary objective is to identify Nash equilibria and optimize defense strategies against various attack patterns. The state space consists of network configurations, the action space includes attack and defense strategies and the reward function measures the impact of attacks and defenses, defined as the sum of the CVSS scores:

$$R = \sum_{t=1}^{T} r_{\pi_t} - \sum_{t=1}^{T} r_{\pi^*}$$

where r_{π_t} is the reward achieved by following policy π_t and r_{π^*} is the reward from the optimal policy π^*. This approach treats each population as an agent with mixed strategies.

Caminero et al. (2019) incorporate a multi-agent technique by integrating a classifier, acting as the agent, with a simulated environment. This environment generates network traffic samples (states) and provides rewards based on the classifier's predictive accuracy. The classifier's objective is to predict (action) the correct intrusion label for the given network samples, while the environment's goal is to actively increase the difficulty of predictions by behaving adversarially, challenging the classifier to learn from the most difficult cases. The **reward function** associates a numeric value with the correctness of the classifier's predictions:

$$R = \begin{cases} 1 & \text{if prediction is correct} \\ 0 & \text{if prediction is incorrect} \end{cases}$$

By maximizing rewards obtained from the environment, the classifier learns to adapt to these challenges, leading to enhanced performance in detecting and classifying intrusions within network traffic. This dynamic interaction between the classifier and the adversarial environment forms the core of Adversarial Environment using RL, enabling it to effectively address the evolving threats and complexities in network security.

Piplai et al. (2022) present a two-player game-based RL environment, utilizing both attacker and defender agents trained simultaneously. The simulation environment is powered by CyberBattleSim, focusing on detecting and mitigating multi-vector and zero-day cyber attacks. The principal features of the approach include the use of cybersecurity knowledge graphs (CKGs) to generate the simulation environment and guide the RL agents. This method reduces the human effort required to explicitly define network vulnerabilities and their impacts. The RL model employs a minimax Deep Q-Network (DQN) for both attacker and defender agents, enhancing the learning process through guided exploration based on expert knowledge. The **reward function** $r(s, a_{\text{attack}}, b_{\text{defend}})$ represents the immediate reward received after taking **actions** a_{attack} and b_{defend} in state s. This immediate reward is designed to capture the success or failure of the agents' actions and their impact on the network's state.

Apruzzese et al. (2020) focus on using DRL to generate realistic adversarial samples that can evade botnet detectors based on network flow analysis. The proposed approach integrates two DRL Frameworks: 2DQN and Sarsa, where the agent learns how to perturb specific traffic features to evade detection. The main attack scenario involves modifying traffic features such as duration, bytes sent and packet count, aiming to evade detection by machine learning-based intrusion detection systems (IDS). The core components of the DRL setup include the **action**, which involves modifying the features of malicious network flows and the **state**, representing the current characteristics of the network traffic. The **policy** defines how the agent selects the next action based on the current state, with the goal of maximizing the **reward**, which is positive if the sample evades detection. The reward function for the 2DQN agent is defined as:

$$Y_t^{\text{2DQN}} = R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a Q(S_{t+1}, a; \theta_t^+); \theta_t^-)$$

where R_{t+1} represents the immediate reward, S_{t+1} is the next state and γ is the discount factor. This method enables the creation of botnet detectors that are resilient against novel and unforeseen evasion attempts while maintaining detection performance in non-adversarial scenarios.

Anderson et al. (2018) introduce an adversarial attack on machine learning-based static Portable Executable (PE) malware detectors using RL. The core idea is to treat malware evasion as a Markov Decision Process, where an RL agent learns functionality-preserving mutations to PE files that evade detection. The proposed method employs an ACER (Actor-Critic with Experience Replay) model to perform black-box evasion attacks. The agent is equipped with a set of operations, such as packing or renaming sections, that it may perform on the PE file, with the goal of evading detection by a machine learning classifier. The features used by the target classifier, a gradient-boosted decision tree model, include PE header metadata, section metadata, import/export table data, byte histograms and entropy histograms, summarized in a 2350-dimensional feature vector representing the **state** of the environment. The **action space** available to the agent involves modifications to the PE file, such as appending bytes, modifying headers, or adding unused functions. The agent's policy is designed to select actions that maximize a binary **reward function**, where:

$$R(s_t, a_t) = \begin{cases} 10, & \text{if the malware evades detection} \\ 0, & \text{if detection occurs} \end{cases}$$

Through a series of games played against the classifier, the agent learns which sequences of operations are likely to result in evasion for any given malware sample. The perturbed samples that successfully bypassed the classifier were uploaded to VirusTotal, a widely-used online service that aggregates and scans files, using more than 65 different anti-malware products to detect viruses, worms, trojans and other kinds of malicious content. The results showed that these perturbed samples were detected as malicious by 50% of the anti-malware products that had detected the original unmodified samples, suggesting that the attack has some real-world implications.

In Shashkov et al. (2023) adversarial reinforcement learning is discussed within the context of cybersecurity, specifically addressing simulations like CyberBattleSim and SNAPT, which are focused on network attacks such as Advanced Persistent Threats (APTs). The **action space** for the attacker includes selecting nodes to exploit (e.g., lateral movement), while the defender can take actions such as reimaging to restore nodes. Policies for both attackers and defenders are optimized using DRL algorithms like Advantage Actor-Critic (A2C) or hybrid approaches that combine DRL and ES. The **state** features the network's security state, including compromised or secure nodes. The agents' **observations** are as follows: attackers see vulnerabilities in connected devices; defenders observe node states and past exploits. The policy is learnt by a neural network (A2C, ES) that guides the selection of actions (exploits or reimaging). The environment is simulation-based (e.g., CyberBattleSim), with a mix of simulated and real-network-inspired architectures. The **reward function** R for the attacker is defined as the sum of the values of all compromised nodes:

$$R = \sum_{i \in C} v_i$$

where C represents the set of compromised nodes and v_i denotes the value of node i. The defender's reward is the negative of the attacker's reward.

Elderman et al. (2017) investigate adversarial interactions between an attacker and a defender in a simulated network environment. This problem is modeled as a Markov game with incomplete information, where both agents – attacker and defender – use RL to optimize their

Table 7
Summary of adversarial RL for Network-based security.

Ref.	Algorithm	Threats	Key metrics	Reward elements
Chowdhary et al. (2021)	Deep-Q Learning	Adaptive adversaries in SDN	Defender's utility, attack complexity	CVSS scores, Attack and defense costs
Novaes et al. (2021)	Deep-Q Learning	DDoS attacks in SDN	Attack detection accuracy, false positives	Attack and defense costs,
Turner et al. (2022)	Advantage Actor-Critic	Breach and Attack Simulation	Nash equilibria, defense strategy	Sum of CVSS scores for affected software settings
Caminero et al. (2019)	DQN and DDQN	General Intrusion Detection	Prediction accuracy, robustness	Prediction
Piplai et al. (2022)	Minimax Deep Q-Network	Multi-vector, zero-day attacks	Cumulative Reward, Steps to Goal, Net. status	Joint optimization, attacker and defender rewards
Apruzzese et al. (2020)	2DQN and Sarsa	adv. evasion attacks	Duration, Bytes Sent	Discount factor
Anderson et al. (2018)	ACER	Portable Executable malware	Malware sample benign or malicious	Binary based on success or failure
Shashkov et al. (2023)	A2C with ES	Network Attacks in CyberBattleSim	Number of compromised nodes	Minimizing compromised nodes violated
Elderman et al. (2017)	MC and Q-learning	Network intrusion attack	Zero-sum game	Binary based on success or failure
Kunz et al. (2022)	PPO, A2C	Network Attacks in CyberBattleSim	Detection Accuracy, Epoch Length	Defenders reward opposite of the attacker

strategies. The attacker attempts to breach network security by hacking nodes, while the defender strengthens the defense mechanisms and attempts to detect intrusions. The **state** of the attacker is defined by the node they occupy in the network and the attacker's **actions** include selecting a neighboring node to attack with a particular type of hacking strategy. On the other hand, the defender's state is a representation of the entire network's defense status, with actions aimed at enhancing defense or detection capabilities at specific nodes. Both agents follow a policy that maps states to actions and their behaviors are driven by RL.

The **reward function** is defined as follows:

$$R = \begin{cases} +100, & \text{if the agent wins (successful attack or successful defense)} \\ -100, & \text{if the agent loses (failed attack or failed defense)} \end{cases}$$

Monte Carlo learning with the Softmax exploration strategy proved to be the most effective for both attacker and defender, particularly in adapting to evolving strategies.

Kunz et al. (2022) explore the application of RL to develop autonomous cyber operation agents in a simulated network environment using CyberBattleSim. The focus is on training red (attacker) and blue (defender) agents to automate the detection and mitigation of network attacks. The paper extends CyberBattleSim to support blue agent training and reports on the results of training blue agents both in isolation and jointly with red agents. Red agent exploits a vulnerability on a node to obtain credentials and a vulnerability on a remote node to gain access and use credentials to take over a remote node. Blue agent can reset a node, removing any attacker control, use a firewall rule to block specific traffic on a node, allow specific traffic on a node, stop a running service on a node and/or start a stopped service on a node (**actions**). The **observation** space includes newly discovered nodes, lateral move successes and known credentials for red agents; for blue agents, it includes infected nodes, firewall rule statuses and service statuses. Training involves decentralized MARL using algorithms like PPO and A2C, implemented via the stable-baselines3 library. For attackers, the reward R_t at time step t is defined by the action type, result and node value. For defenders, the reward is the negation of the attacker's reward, with an additional penalty if network availability constraints are violated. The **reward function** for the defender agent is defined as:

$$R_t^{\text{defender}} = -R_t^{\text{attacker}} \tag{5}$$

where R_t^{attacker} is the reward received by the attacker at time step t.

Discussion. In summary, adversarial RL Frameworks for network-based security (see Table 7) provide an innovative solution to train to counteract sophisticated cyber threats. For instance, attackers may learn to poison training data or influence the environment in ways that steer the (defending) learning agent toward suboptimal or harmful policies, thus undermining the system's effectiveness in defending against cyber threats (Kunz et al., 2022; Piplai et al., 2022; Rezapour & Tzeng, 2022). For example the paper (Kunz et al., 2022), highlights challenges such as over-specialization of jointly trained agents. Common modeling techniques include zero-sum games and competitive multi-agent settings to simulate real-world attacker-defender interactions, often using algorithms like Deep Q-learning and policy-gradient methods like REINFORCE or PPO. Also Generative Adversarial Networks (GANs) are vastly used, to simulate and counteract sophisticated attack strategies. GAN-based adversarial approaches are particularly effective in generating realistic attack scenarios, which can then be used to train robust defense policies. Additionally, adversarial training with PPO has been shown to improve the resilience of SDN controllers against targeted attacks. These studies demonstrate the effectiveness of training RL agents in adversarial environments, where the presence of hostile learning agents enhances the agents' ability to anticipate and respond to malicious activities. Reward functions frequently incorporate elements such as CVSS scores and attack/defense costs, highlighting the need to balance attack mitigation with network performance. This adversarial training framework is crucial for developing robust and resilient cybersecurity systems capable of defending against increasingly complex and evolving threats.

Game-theoretic perspective. Game theory provides robust frameworks for modeling and addressing adversarial cybersecurity challenges. As discussed in the previous paragraph and highlighted in the survey by Nguyen and Reddi (2023), these frameworks can intersect with RL. They enable the analysis of interactions between attackers and defenders as strategic competitive games, facilitating the derivation of optimal policies for both parties. In particular, Nash equilibrium and Stackelberg game models have been widely applied to adversarial RL settings. For example, Nash equilibrium helps to identify the optimal strategies for players in a zero-sum game, where the gain of one player directly corresponds to the loss of the other. Stackelberg games, on the other hand, are leader–follower models where one agent (leader) commits to a strategy before the other (follower) responds. These concepts are crucial for addressing sophisticated and dynamic adversarial threats in cybersecurity systems. Several studies

Table 8
Summary of simulators used.

References	Simulator
Akbari et al. (2020), Dake et al. (2021), Janakiraman and Deva Priya (2023), Liu et al. (2018), Novaes et al. (2021), Rezapour and Tzeng (2022), Simpson et al. (2020) and Yungaicela-Naula et al. (2022)	Mininet (Kaur, Singh, & Ghumman, 2014)
Phan et al. (2020)	MaxiNet (Wette et al., 2014)
Zolotukhin et al. (2020)	OpenStack with the Opendaylight SDN controller (Subramanian & Voruganti, 2016)
Anderson et al. (2018) and Chowdhary et al. (2021)	OpenAI Gym (Brockman et al., 2016)
Sethi et al. (2021)	Network Simulator - ns-2 (Simulator, 2009)
Feng et al. (2020)	Application Layer DDoS Simulator
Kunz et al. (2022), Piplai et al. (2022) and Shashkov et al. (2023)	CyberBattleSim (Team, 2021)

have utilized these approaches: Addesso, Barni, Di Mauro, and Matta (2021) employed Kendall's model with adversarial game theory to develop a containment strategy for distributed cyber-threats. This work combines stochastic processes and Nash equilibria to identify effective countermeasures. Zhang, Zhu, Hussain, Ye and Zhou (2023) proposed a game-theoretic method to defend against advanced persistent threats (APTs), balancing resource management and timing to enhance system resilience. Chivukula, Yang, Liu, Zhu, and Zhou (2021) introduced a variational adversarial learning framework, leveraging Nash equilibrium for adversarial deep learning in cybersecurity. Lian, Jia, Wu, and Huang (2023) used a Stackelberg game to analyze the stability of networked systems under denial-of-service (DoS) attacks, offering a novel approach to ensuring system robustness. Addesso, Cirillo, Di Mauro, and Matta (2019) explored adversarial game theory in the context of networking, focusing on the detection of encrypted and concealed VoIP traffic. By integrating game-theoretic principles with adversarial signal processing, this study advanced secure communication systems. These studies highlight the importance of game theory in adversarial RL for cybersecurity, providing valuable insights into defense mechanisms against evolving threats.

5.6. Simulation software and open datasets used

The reviewed studies demonstrate a diverse range of (possibly deep, multi-agent) RL applications in cybersecurity, spanning various network configurations and agent setups. Most approaches are evaluated through simulations (see Table 8), using emulators such as Mininet, MaxiNet and others. Mininet is the most commonly used emulator, offering a realistic network environment that allows researchers to simulate complex SDN topologies and evaluate the effectiveness of RL-based security measures. MaxiNet extends Mininet's capabilities by supporting distributed network emulation, enabling larger and more scalable experiments.

Virtualized network environments integrated with platforms like Gynmnasium (formerly known as OpenAI Gym), or OpenStack provide flexibility and easy integration for developing and testing RL algorithms, particularly in the context of SDN and cloud environments. CyberBattleSim, specifically designed for cybersecurity research, facilitates the simulation of network attacks and defense mechanisms, providing a robust test-bed for adversarial RL studies. The reliance on simulated environments highlights the challenges in deploying RL-based security solutions in real-world settings. Future research should focus on bridging this gap by implementing and testing these approaches in operational networks, ensuring their robustness and effectiveness against real-world cyber threats.

Table 9 provides a list of the open datasets identified in the reviewed studies, which have been used in the application of RL for various cybersecurity tasks, including malware detection, intrusion detection, DDoS attack mitigation and IoT security. These datasets play a crucial role in evaluating the effectiveness of RL-based methods in real-world scenarios, offering a standardized framework for benchmarking algorithms and facilitating comparison of results across different studies. The use of diverse datasets also highlights the wide range of challenges that RL Frameworks address in cybersecurity, from detecting advanced persistent threats to mitigating network-based attacks.

6. Open challenges

While RL holds great promise for addressing cybersecurity challenges, it still faces several open challenges, as summarized in Table 10. Most of these challenges are intrinsic to RL methods, but they are particularly exacerbated and becomes even more crucial to deal with in cybersecurity environments.

6.1. Real-world deployments

ML techniques are widely used in cybersecurity for tasks such as intrusion detection, anomaly detection and malware classification. Traditional ML methods, such as supervised learning and clustering, have achieved high accuracy when trained on datasets like CICIDS2017 (Sharafaldin et al., 2018) and UNSW-NB15 (Moustafa & Slay, 2015). These methods excel in identifying known attack patterns from static data and have been successfully deployed in real-world scenarios, such as commercial IDS and threat intelligence platforms (Cisco Secure IPS, FireEye Helix, McAfee Network Security Platform, etc.). However, the rigidity of these network architectures and the lack of centralized control make it challenging to implement adaptive defenses, particularly against evolving threats. In contrast, SDNs offer centralized control, programmability and global visibility, making them fertile ground for RL-based approaches (Elnaggar & Bezzo, 2018). RL agents in SDNs have shown promise in mitigating threats such as DDoS attacks (Feng et al., 2020) by dynamically rerouting traffic or rate-limiting malicious flows. However, RL adoption in both traditional and SDN environments faces unique challenges, such as scalability (Xu & Luo, 2007), adversarial robustness (Oh & Iyengar, 2019) and integration with legacy systems (Xu & Xie, 2005).

6.2. Complexity and scalability

One of the primary challenges of RL in cybersecurity is *complexity*. Cybersecurity environments are often characterized by high-dimensional and dynamic conditions, making the training process computationally expensive. This complexity arises from the need to accurately represent diverse network states, attacker behaviors and defensive actions. As a result, RL algorithms may encounter *scalability* issues, leading to prolonged training times and resource constraints, which limit their practical applicability in real-world settings (Liu et al., 2018; Phan et al., 2019, 2020). Traditional networks are often characterized by decentralized control and fixed routing mechanisms, which pose significant challenges for scaling RL solutions. For instance, the distributed nature of traffic management increases computational overhead, as RL agents need to monitor and act on individual devices. SDNs, while more centralized, introduce complexity due to the large-scale, dynamic nature of flows managed by the controller. In SDN environments, high-dimensional state spaces and frequent state changes exacerbate scalability issues, requiring optimized RL algorithms tailored for dynamic flow control (Liu et al., 2018; Phan et al., 2019, 2020).

Table 9
Summary of open datasets used and their applications.

Ref.	Dataset name	Source	Application	Details
Elnaggar and Bezzo (2018)	ISCX 2012	CIC/UNB (CIC/UNB, 2012)	Predict and recover from cyber–physical attacks on UAVs	Intrusion Detection data for anomaly detection
Caminero et al. (2019), Sethi et al. (2021) and Veluchamy and Kathavarayan (2022)	NSL-KDD	CIC/UNB (CIC/UNB, 2000)	Network Intrusion Detection	Improved version of the KDD'99
Bhosale et al. (2014) and Simpson et al. (2020)	DARPA KDD99	MIT Lincoln lab. (Laboratory, 1999)	Intrusion Detection Evaluation	Network traffic and audit logs collected on a simulation network
Dasgupta et al. (2022)	Honda Driving	Honda Research Institute Driving (Ramanishka, Chen, Misu, & Saenko, 2018)	Real human driving	104 h of real human driving in the San Francisco Bay Area collected using different sensors
Piplai et al. (2022)	CDX	NSA Red Team (Team, 2017)	Attacks and confidentiality/integrity	Packets transmitted within the team's subnet during the periods the sensor was active
Apruzzese et al. (2020)	CTU-13	Stratosphere Lab	Botnet Detection	Network traffic from different botnets
Yungaicela-Naula et al. (2022) and Zhang, Qiu et al. (2023)	CICIDS 2017	CIC/UNB (CIC/UNB, 2017)	DDoS Attack Mitigation	Realistic network traffic for IDS
Janakiraman and Deva Priya (2023)	Hogzilla Dataset	Hogzilla IDS (IDS, 2017)	DDoS Attack Mitigation	Data from DDoS and other attacks
Oh and Iyengar (2019)	GeoLife GPS	Microsoft Research Asia (Zheng, Fu, Xie, Ma, & Li, 2011)	GPS	Trajectories collected from individual users over three years.
Oh and Iyengar (2019)	TST	ECML-PKDD (ECML-PKDD, 2014)	GPS	Trajectories collected from taxi drivers in the city of Porto

Table 10
Open challenges identified in RL studies.

Ref.	Challenge	Description
Oh and Iyengar (2019), Xu and Luo (2007) and Xu and Xie (2005)	Real-World deployments	Include integrating RL into rigid legacy systems and ensuring scalability and adversarial robustness.
Liu et al. (2018), Phan et al. (2019) and Phan et al. (2020)	Complexity and scalability	Issues handling large-scale networks or high-volume traffic.
Simpson et al. (2020), Turner et al. (2022) and Zhang, Qiu et al. (2023)	Reward function specification	Challenges in defining reward functions that balance multiple objectives in security tasks.
Guo et al. (2020), Phan et al. (2019) and Zolotukhin et al. (2020)	Sample efficiency	Requires large amounts of data for effective training, often impractical in cybersecurity.
Chowdhary et al. (2021) and Liu et al. (2018)	Computation overhead	High computational requirements for training and executing RL algorithms, limiting their scalability.
Caminero et al. (2019) and Phan et al. (2020)	Data quality	Reliance on high-quality, diverse datasets for training, posing challenges in realistic cybersecurity scenarios.
Akbari et al. (2020) and Kim et al. (2021)	Generalization	Difficulty adapting to rapidly changing network states, leading to poor generalization.
Caminero et al. (2019) and Phan et al. (2020)	Transfer learning	Applying knowledge from one security context to another can enhance efficiency but remains under-explored.
Simpson et al. (2020) and Zhang, Qiu et al. (2023)	Explainability	RL models' decisions need to be interpretable and understandable for stakeholders.

6.3. Reward function specification

One of the key challenges in applying RL to cybersecurity is the design of an appropriate reward function. The reward function needs to encapsulate the multiple objectives of a cybersecurity system, balancing between false positives, true detections and timely responses to threats. A poorly designed reward function may lead the RL agent to focus on suboptimal behaviors, such as avoiding certain complex but necessary tasks. It is essential that the reward function accurately reflects the overall goals of the cybersecurity strategy to ensure that the RL agent's learning is aligned with effective defense (Simpson et al., 2020; Turner et al., 2022; Zhang, Qiu et al., 2023). In traditional networks, designing an RL reward function involves balancing conflicting goals, such as reducing latency while ensuring attack detection. These networks often lack real-time feedback mechanisms, making reward calibration more challenging. SDNs offer more opportunities to design dynamic reward functions based on global network metrics, such as flow statistics and topology changes. However, the need to align these metrics with cybersecurity objectives, such as minimizing collateral damage during mitigation, remains a critical challenge (Simpson et al., 2020; Turner et al., 2022; Zhang, Qiu et al., 2023).

6.4. Sample efficiency

RL algorithms generally require large amounts of data to learn effective policies. In cybersecurity environments, collecting such vast, high-quality data is not so easy, particularly in the case of detecting zero-day attacks, where labeled data is sparse or non-existent.

Improving sample efficiency is vital for enhancing the practicality of RL systems in cybersecurity (Guo et al., 2020; Phan et al., 2019; Zolotukhin et al., 2020). Traditional networks suffer from data sparsity due to limited centralized visibility, making it difficult to train RL models effectively. This is especially true for detecting zero-day attacks, where labeled data is rarely available. SDNs alleviate some of these issues by providing global state data through the controller, enabling more efficient training. However, the scale and complexity of SDN data require novel methods to improve sample efficiency without overwhelming computational resources (Guo et al., 2020; Phan et al., 2019; Zolotukhin et al., 2020).

6.5. Computation overhead

RL algorithms, especially in complex network environments, require significant computational resources for training and deployment. The process of exploration and exploitation, fundamental to RL, often results in long training times, especially when working with high-dimensional state spaces or large-scale networks. This high computational overhead can limit the scalability and practical deployment of RL Frameworks in resource-constrained environments, such as those with limited processing power or bandwidth (Chowdhary et al., 2021; Liu et al., 2018). In traditional networks, distributed agents require significant resources for local decision-making, adding to computational overhead. This decentralized approach contrasts with SDNs, where RL models can use centralized resources at the controller level. Despite this advantage, high-dimensional action spaces in SDNs still pose significant challenges to achieve real-time responses (Chowdhary et al., 2021; Liu et al., 2018).

6.6. Data quality and high fidelity simulation

The effectiveness of RL frameworks in cybersecurity is heavily dependent on the quality of the training data. High-quality and diverse datasets are crucial for training RL models that can generalize to various attack vectors and network conditions. However, in many real-world cybersecurity applications, obtaining large, representative datasets can be difficult due to privacy concerns, the proprietary nature of data, or the unpredictability of zero-day attacks. Poor data quality can result in RL models that fail to detect certain types of threats or perform inconsistently across different network environments (Caminero et al., 2019; Phan et al., 2020). In traditional networks, the decentralized nature of monitoring systems often leads to fragmented and inconsistent datasets, which undermine RL performance. SDNs, with their centralized architecture, offer more structured and comprehensive data. However, ensuring the quality, diversity and representativeness of these data, especially in dynamic cybersecurity scenarios, remains a critical bottleneck for both environments. Given that RL is often trained online, simulators play a key role in this process, providing safe and controlled environments to test and refine agents before deployment. These simulators can help overcome limitations of real-world data by generating diverse and scalable scenarios, enabling RL agents to learn robust policies under various network conditions and attack vectors.

6.7. Generalization and transfer learning

The challenge of *generalization* refers to the RL model's ability to perform well on new data or in different network conditions compared to the environment it was originally trained on. An RL agent trained to detect specific types of threats should be able to generalize to different types of threats or other security challenges, even in varying network environments, but this is still difficult to achieve reliably (Akbari et al., 2020; Kim et al., 2021).

One promising area of ongoing research is *transfer learning*, where knowledge acquired by RL agents in one context is applied to another. For example, an RL agent trained to detect malware could potentially transfer its learned policies to enhance intrusion detection in network traffic. Transfer learning has the potential to significantly improve the efficiency of RL-based systems by reducing the need to train agents from scratch for every new task or environment (Caminero et al., 2019; Phan et al., 2020).

6.8. Explainability

The *explainability* of RL systems remains an open challenge, especially in high-stakes cybersecurity applications. It is critical that the actions of RL algorithms are interpretable and understandable to human stakeholders. For instance, security analysts must be able to trust and comprehend the decisions made by RL-based autonomous threat response systems, ensuring that these systems can be effectively deployed in real-world scenarios (Simpson et al., 2020; Zhang, Qiu et al., 2023).

7. Conclusion & future works

In this paper, we have discussed the potential of reinforcement learning (RL) to enhance cybersecurity defenses by offering adaptive and dynamic mechanisms for detection, mitigation and response capabilities against sophisticated cyber threats. The studies reviewed illustrate the versatility of RL algorithms across different network configurations, including single-agent and multi-agent systems, as well as adversarial settings. We highlighted the limitations of traditional approaches, explained why RL can help surpass them and proposed a bi-dimensional classification to provide researchers with a comprehensive overview as a starting point in the field.

While our analysis demonstrates the promise of RL in improving detection, mitigation and response capabilities, the surveyed literature also identifies several challenges that must be addressed. These include model complexity, sample efficiency, generalization capabilities and others. Despite the advancements made in our survey, there are still areas for potential improvement. For example, while we have provided a detailed categorization of RL Frameworks, further work could focus on developing standardized benchmarks for evaluating the performance of these techniques in various cybersecurity scenarios. Additionally, incorporating more real-world case studies and experimental validations could strengthen the practical relevance of the survey.

Moving forward, it is imperative to develop robust and explainable RL-based defense mechanisms, as well as explore techniques for knowledge transfer and generalization across diverse cyber threats and environments. By addressing these challenges, we can harness the full potential of RL to fortify cybersecurity defenses and effectively mitigate emerging threats.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

Adawadkar, A. M. K., & Kulkarni, N. (2022). Cyber-security and reinforcement learning — A brief survey. *Engineering Applications of Artificial Intelligence*, [ISSN: 0952-1976] *114*, Article 105116. http://dx.doi.org/10.1016/j.engappai.2022.105116.

Addesso, P., Barni, M., Di Mauro, M., & Matta, V. (2021). Adversarial Kendall's model towards containment of distributed cyber-threats. *IEEE Transactions on Information Forensics and Security*, *16*, 3604–3619. http://dx.doi.org/10.1109/TIFS.2021.3082327.

Addesso, P., Cirillo, M., Di Mauro, M., & Matta, V. (2019). Advoip: Adversarial detection of encrypted and concealed voip. *IEEE Transactions on Information Forensics and Security*, *15*, 943–958.

Akbari, I., Tahoun, E., Salahuddin, M. A., Limam, N., & Boutaba, R. (2020). ATMoS: Autonomous threat mitigation in SDN using reinforcement learning. In *NOMS 2020 - 2020 IEEE/iFIP network operations and management symposium* (pp. 1–9). http://dx.doi.org/10.1109/NOMS47738.2020.9110426.

Anderson, H. S., Kharkar, A., Filar, B., Evans, D., & Roth, P. (2018). Learning to evade static pe machine learning malware models via reinforcement learning. arXiv preprint arXiv:1801.08917.

Apruzzese, G., Andreolini, M., Marchetti, M., andrea, V., & Colajanni, M. (2020). Deep reinforcement adversarial learning against botnet evasion attacks. *IEEE Transactions on Network and Service Management*, *17*(4), 1975–1987. http://dx.doi.org/10.1109/TNSM.2020.3031843.

Bhagyashree Deokar, A. H. (2012). Intrusion detection system using log files and reinforcement learning. *International Journal of Computer Applications*, [ISSN: 0975-8887] *45*(19), 28–35. http://dx.doi.org/10.5120/7026-9675.

Bhosale, R., Mahajan, S., & Kulkarni, P. (2014). Cooperative machine learning for intrusion detection system. *International Journal of Scientific and Engineering Research*, *5*(1), 1780–1785.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., et al. (2016). Openai gym. arXiv preprint arXiv:1606.01540.

Caminero, G., Lopez-Martin, M., & Carro, B. (2019). Adversarial environment reinforcement learning algorithm for intrusion detection. *Computer Networks*, [ISSN: 1389-1286] *159*, 96–109. http://dx.doi.org/10.1016/j.comnet.2019.05.013.

Cengiz, E., & Gök, M. (2023). Reinforcement learning applications in cyber security: A review. *Sakarya University Journal of Science*, *27*(2), 481–503. http://dx.doi.org/10.16984/saufenbilder.1237742.

Chivukula, A. S., Yang, X., Liu, W., Zhu, T., & Zhou, W. (2021). Game theoretical adversarial deep learning with variational adversaries. *IEEE Transactions on Knowledge and Data Engineering*, *33*(11), 3568–3581. http://dx.doi.org/10.1109/TKDE.2020.2972320.

Chowdhary, A., Huang, D., Sabur, A., Vadnere, N., Kang, M., & Montrose, B. (2021). SDN-based moving target defense using multi-agent reinforcement learning. In *Proceedings of first international conference on autonomous intelligent cyber defense agents* (p. 100).

CIC/UNB (2000). Dataset NSL-KDD. URL: https://www.kaggle.com/datasets/hassan06/nslkdd.

CIC/UNB (2012). Dataset ISCX 2012. URL: https://www.unb.ca/cic/datasets/ids.html.

CIC/UNB (2017). Dataset CICIDS 2017. URL: https://www.unb.ca/cic/datasets/ids-2017.html.

Connell, J. H., & Sridhar Mahadevan, K. (1999). Robot learning. *Robotica*, *17*(2), 229–235. http://dx.doi.org/10.1017/S0263574799271172.

Dake, D. K., Gadze, J. D., Klogo, G. S., & Nunoo-Mensah, H. (2021). Multi-agent reinforcement learning framework in SDN-IoT for transient load detection and prevention. *Technologies*, [ISSN: 2227-7080] *9*(3), http://dx.doi.org/10.3390/technologies9030044.

Dasgupta, S., Ghosh, T., & Rahman, M. (2022). A reinforcement learning approach for global navigation satellite system spoofing attack detection in autonomous vehicles. *Transportation Research Record*, *2676*(12), 318–330. http://dx.doi.org/10.1177/03611981221095509.

Denning, D. (1987). An intrusion-detection model. *IEEE Transactions on Software Engineering*, *SE-13*(2), 222–232. http://dx.doi.org/10.1109/TSE.1987.232894.

Dixit, P., & Silakari, S. (2021). Deep learning algorithms for cybersecurity applications: A technological and status review. *Computer Science Review*, [ISSN: 1574-0137] *39*, Article 100317. http://dx.doi.org/10.1016/j.cosrev.2020.100317.

ECML-PKDD (2014). Dataset TST. URL: https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i/data.

Elderman, R., Pater, L. J., Thie, A. S., Drugan, M. M., & Wiering, M. A. (2017). Adversarial reinforcement learning in a cyber security simulation. In *9th international conference on agents and artificial intelligence* (pp. 559–566). SciTePress Digital Library.

Elnaggar, M., & Bezzo, N. (2018). An IRL approach for cyber-physical attack intention prediction and recovery. In *2018 annual American control conference* (pp. 222–227). http://dx.doi.org/10.23919/ACC.2018.8430922.

Feng, Y., Li, J., & Nguyen, T. (2020). Application-layer ddos defense with reinforcement learning. In *IEEE/ACM 28th international symposium on quality of service* (pp. 1–10). http://dx.doi.org/10.1109/IWQoS49365.2020.9213026.

Finistrella, S., Mariani, S., & Zambonelli, F. (2024). Multi-agent reinforcement learning for cybersecurity: Approaches and challenges. In *CEUR workshop proceedings: Vol. 3735, Proceedings of the 25th workshop "from objects to agents", Bard (Aosta), Italy, July 8-10, 2024* (pp. 103–118). CEUR-WS.org.

Fischer, T. G. (2018). *Reinforcement learning in financial markets-a survey: Technical report*, FAU Discussion Papers in Economics.

Fragkos, G., Johnson, J., & Tsiropoulou, E. E. (2022). Dynamic role-based access control policy for smart grid applications: An offline deep reinforcement learning approach. *IEEE Transactions on Human-Machine Systems*, *52*(4), 761–773. http://dx.doi.org/10.1109/THMS.2022.3163185.

Gulmez, H., & Angin, P. (2020). A study on the efficacy of deep reinforcement learning for intrusion detection. *Sakarya University Journal of Computer and Information Sciences*, *4*, http://dx.doi.org/10.35377/saucis.04.01.834048.

Guo, Y. (2023). A review of machine learning-based zero-day attack detection: Challenges and future directions. *Computer Communications*, [ISSN: 0140-3664] *198*, 175–185. http://dx.doi.org/10.1016/j.comcom.2022.11.001.

Guo, X., Lin, H., Li, Z., & Peng, M. (2020). Deep-reinforcement-learning-based qos-aware secure routing for SDN-IoT. *IEEE Internet of Things Journal*, *7*(7), 6242–6251. http://dx.doi.org/10.1109/JIOT.2019.2960033.

He, H., Maple, C., Watson, T., Tiwari, A., Mehnen, J., Jin, Y., et al. (2016). The security challenges in the IoT enabled cyber-physical systems and opportunities for evolutionary computing & other computational intelligence. In *2016 IEEE congress on evolutionary computation* (pp. 1015–1021). IEEE.

Hu, Z., Chen, P., Zhu, M., & Liu, P. (2019). Reinforcement learning for adaptive cyber defense against zero-day attacks. In *Adversarial and uncertain reasoning for adaptive cyber defense: control- and game-theoretic approaches to cyber security* (pp. 54–93). Cham: Springer International Publishing, ISBN: 978-3-030-30719-6, http://dx.doi.org/10.1007/978-3-030-30719-6_4.

IDS, H. (2017). Dataset hogzilla. URL: https://ids-hogzilla.org/dataset/.

Janakiraman, S., & Deva Priya, M. (2023). A deep reinforcement learning-based ddos attack mitigation scheme for securing big data in fog-assisted cloud environment. *Wireless Personal Communications*, *130*(4), 2869–2886. http://dx.doi.org/10.1007/s11277-023-10407-2.

Kaur, K., Singh, J., & Ghumman, N. S. (2014). Mininet as software defined networking testing platform. In *International conference on communication, computing & systems* (pp. 139–142). IEEE.

Kim, S., Yoon, S., & Lim, H. (2021). Deep reinforcement learning-based traffic sampling for multiple traffic analyzers on software-defined networks. *IEEE Access*, *9*, 47815–47827. http://dx.doi.org/10.1109/ACCESS.2021.3068459.

Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, *32*(11), 1238–1274. http://dx.doi.org/10.1177/0278364913495721.

Kunz, T., Fisher, C., Novara-Gsell, J. L., Nguyen, C., & Li, L. (2022). A multiagent CyberBattleSim for RL cyber operation agents. In *2022 international conference on computational science and computational intelligence* (pp. 897–903). http://dx.doi.org/10.1109/CSCI58124.2022.00161.

Laboratory, M. L. (1999). Dataset DARPA-KDD99. URL: https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset.

Lian, J., Jia, P., Wu, F., & Huang, X. (2023). A stackelberg game approach to the stability of networked switched systems under DoS attacks. *IEEE Transactions on Network Science and Engineering*, *10*(4), 2086–2097. http://dx.doi.org/10.1109/TNSE.2023.3240687.

Liu, Y., Dong, M., Ota, K., Li, J., & Wu, J. (2018). Deep reinforcement learning based smart mitigation of ddos flooding in software-defined networks. In *IEEE 23rd international workshop on computer aided modeling and design of communication links and networks* (pp. 1–6). http://dx.doi.org/10.1109/CAMAD.2018.8514971.

Liu, J., Xiao, L., Liu, G., & Zhao, Y. (2017). Active authentication with reinforcement learning based on ambient radio signals. *Multimedia Tools and Applications*, [ISSN: 1573-7721] *76*(3), 3979–3998. http://dx.doi.org/10.1007/s11042-015-2958-x.

Macas, M., Wu, C., & Fuertes, W. (2022). A survey on deep learning for cybersecurity: Progress, challenges and opportunities. *Computer Networks*, *212*, Article 109032. http://dx.doi.org/10.1016/j.comnet.2022.109032.

Malialis, K., & Kudenko, D. (2015). Distributed response to network intrusions using multiagent reinforcement learning. *Engineering Applications of Artificial Intelligence*, [ISSN: 0952-1976] *41*, 270–284. http://dx.doi.org/10.1016/j.engappai.2015.01.013.

Martínez Torres, J., Iglesias Comesaña, C., & García-Nieto, P. J. (2019). Machine learning techniques applied to cybersecurity. *International Journal of Machine Learning and Cybernetics*, *10*(10), 2823–2836.

Mishra, S., Sagban, R., Yakoob, A., & Gandhi, N. (2021). Swarm intelligence in anomaly detection systems: an overview. *International Journal of Computers and Applications*, *43*(2), 109–118.

Morgan, S. (2022). Cybercrime to cost the world 8 trillion annually in 2023.

Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 military communications and information systems conference* (pp. 1–6). IEEE.

Nguyen, N. D., Nguyen, T. T., Nguyen, H., & Nahavandi, S. (2020). Review, analyze and design a comprehensive deep reinforcement learning framework. CoRR abs/2002.11883. arXiv:2002.11883. URL: https://arxiv.org/abs/2002.11883.

Nguyen, T. T., & Reddi, V. J. (2023). Deep reinforcement learning for cyber security. *IEEE Transactions on Neural Networks and Learning Systems*, *34*(8), 3779–3795. http://dx.doi.org/10.1109/TNNLS.2021.3121870.

Novaes, M. P., Carvalho, L. F., Lloret, J., & Proença, M. L. (2021). Adversarial deep learning approach detection and defense against DDoS attacks in SDN environments. *Future Generation Computer Systems*, [ISSN: 0167-739X] *125*, 156–167. http://dx.doi.org/10.1016/j.future.2021.06.047.

Oh, M.-h., & Iyengar, G. (2019). Sequential anomaly detection using inverse reinforcement learning. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1480–1490). Association for Computing Machinery, ISBN: 9781450362016, http://dx.doi.org/10.1145/3292500.3330932.

Ozkan-Okay, M., Akin, E., Aslan, Ö., Kosunalp, S., Iliev, T., Stoyanov, I., et al. (2024). A comprehensive survey: Evaluating the efficiency of artificial intelligence and machine learning techniques on cyber security solutions. *IEEe Access*, *12*, 12229–12256.

Phan, T. V., Gias, T. M. R., Islam, S. T., Huong, T. T., Thanh, N. H., & Bauschert, T. (2019). Q-MIND: Defeating stealthy DoS attacks in SDN with a machine-learning based defense framework. In *2019 IEEE global communications conference* (pp. 1–6). http://dx.doi.org/10.1109/GLOBECOM38437.2019.9013585.

Phan, T. V., Nguyen, T. G., Dao, N.-N., Huong, T. T., Thanh, N. H., & Bauschert, T. (2020). DeepGuard: Efficient anomaly detection in SDN with fine-grained traffic flow monitoring. *IEEE Transactions on Network and Service Management*, *17*(3), 1349–1362. http://dx.doi.org/10.1109/TNSM.2020.3004415.

Piplai, A., Anoruo, M., Fasaye, K., Joshi, A., Finin, T., & Ridley, A. (2022). Knowledge guided two-player reinforcement learning for cyber attacks and defenses. In *2022 21st IEEE international conference on machine learning and applications* (pp. 1342–1349). http://dx.doi.org/10.1109/ICMLA55696.2022.00213.

Ramanishka, V., Chen, Y.-T., Misu, T., & Saenko, K. (2018). Toward driving scene understanding: A dataset for learning driver behavior and causal reasoning. In *Conference on computer vision and pattern recognition*.

Rezapour, A., & Tzeng, W.-G. (2022). RL-shield: Mitigating target link-flooding attacks using SDN and deep reinforcement learning routing algorithm. *IEEE Transactions on Dependable and Secure Computing*, *19*(6), 4052–4067. http://dx.doi.org/10.1109/TDSC.2021.3118081.

Sarker, I. H., Kayes, A., Badsha, S., Alqahtani, H., Watters, P., & Ng, A. (2020). Cybersecurity data science: an overview from machine learning perspective. *Journal of Big Data*, *7*, 1–29.

Sethi, K., Madhav, Y. V., Kumar, R., & Bera, P. (2021). Attention based multi-agent intrusion detection systems using reinforcement learning. *Journal of Information Security and Applications*, [ISSN: 2214-2126] *61*, Article 102923. http://dx.doi.org/10.1016/j.jisa.2021.102923.

Sewak, M., Sahay, S. K., & Rathore, H. (2022). Deep reinforcement learning for cybersecurity threat detection and protection: A review. In R. Krishnan, H. R. Rao, S. K. Sahay, S. Samtani, & Z. Zhao (Eds.), *Secure knowledge management in the artificial intelligence era* (pp. 51–72). Cham: Springer International Publishing, ISBN: 978-3-030-97532-6.

Shamshirband, S., Patel, A., Anuar, N. B., Kiah, M. L. M., & Abraham, A. (2014). Cooperative game theoretic approach using fuzzy Q-learning for detecting and preventing intrusions in wireless sensor networks. *Engineering Applications of Artificial Intelligence*, [ISSN: 0952-1976] *32*, 228–241. http://dx.doi.org/10.1016/j.engappai.2014.02.001.

Shao, K., Tang, Z., Zhu, Y., Li, N., & Zhao, D. (2019). A survey of deep reinforcement learning in video games. arXiv:1912.10944.

Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th international conference on information systems security and privacy* (pp. 108–116). SciTePress.

Shashkov, A., Hemberg, E., Tulla, M., & O'Reilly, U.-M. (2023). Adversarial agent-learning for cybersecurity: a comparison of algorithms. *The Knowledge Engineering Review*, *38*.

Shin, S., Xu, L., Hong, S., & Gu, G. (2016). Enhancing network security through software defined networking (SDN). In *25th international conference on computer communication and networks* (pp. 1–9). http://dx.doi.org/10.1109/ICCCN.2016.7568520.

Simpson, K. A., Rogers, S., & Pezaros, D. P. (2020). Per-host ddos mitigation by direct-control reinforcement learning. *IEEE Transactions on Network and Service Management*, *17*(1), 103–117. http://dx.doi.org/10.1109/TNSM.2019.2960202.

Simulator, N. (2009). The network simulator–ns-2. NS-2. Html.

Subramanian, S., & Voruganti, S. (2016). *Software-defined networking (SDN) with OpenStack*. Packt Publishing Ltd.

Team, N. R. (2017). Dataset CDX. URL: https://www.flyn.org/CDX/.

Team, M. (2021). CyberBattleSim. URL: https://github.com/microsoft/cyberbattlesim.

Tripathi, M., Gaur, M. S., & Laxmi, V. (2013). Comparing the impact of black hole and gray hole attack on LEACH in WSN. *Procedia Computer Science*, *19*, 1101–1107.

Turner, M. J., Hemberg, E., & O'Reilly, U.-M. (2022). Analyzing multi-agent reinforcement learning and coevolution in cybersecurity. In *Proceedings of the genetic and evolutionary computation conference* (pp. 1290–1298). New York, NY, USA: Association for Computing Machinery, ISBN: 9781450392372, http://dx.doi.org/10.1145/3512290.3528844.

Uprety, A., & Rawat, D. B. (2021). Reinforcement learning for IoT security: A comprehensive survey. *IEEE Internet of Things Journal*, *8*(11), 8693–8706. http://dx.doi.org/10.1109/JIOT.2020.3040957.

Veluchamy, S., & Kathavarayan, R. S. (2022). Deep reinforcement learning for building honeypots against runtime DoS attack. *International Journal of Intelligent Systems*, *37*(7), 3981–4007. http://dx.doi.org/10.1002/INT.22708.

Wette, P., Dräxler, M., Schwabe, A., Wallaschek, F., Zahraee, M. H., & Karl, H. (2014). Maxinet: Distributed emulation of software-defined networks. In *2014 IFIP networking conference* (pp. 1–9). IEEE.

Xiao, L., Wan, X., Dai, C., Du, X., Chen, X., & Guizani, M. (2018). Security in mobile edge caching with reinforcement learning. *IEEE Wireless Communications*, *25*(3), 116–122. http://dx.doi.org/10.1109/MWC.2018.1700291.

Xin, Y., Kong, L., Liu, Z., Chen, Y., Li, Y., Zhu, H., et al. (2018). Machine learning and deep learning methods for cybersecurity. *IEEE Access*, *6*, 35365–35381.

Xu, X. (2010). Sequential anomaly detection based on temporal-difference learning: Principles, models and case studies. *Applied Soft Computing*, [ISSN: 1568-4946] *10*(3), 859–867. http://dx.doi.org/10.1016/j.asoc.2009.10.003.

Xu, X., & Luo, Y. (2007). A kernel-based reinforcement learning approach to dynamic behavior modeling of intrusion detection. In *Advances in neural networks – ISNN 2007* (pp. 455–464). Springer Berlin Heidelberg, ISBN: 978-3-540-72383-7, http://dx.doi.org/10.1007/978-3-540-72383-7_54.

Xu, X., & Xie, T. (2005). A reinforcement learning approach for host-based intrusion detection using sequences of system calls. In *Advances in intelligent computing* (pp. 995–1003). Berlin, Heidelberg: Springer Berlin Heidelberg, ISBN: 978-3-540-31902-3, http://dx.doi.org/10.1007/11538059_103.

Yaseen, A. (2023). The role of machine learning in network anomaly detection for cybersecurity. *Sage Science Review of Applied Machine Learning*, *6*(8), 16–34.

Yu, C., Liu, J., Nemati, S., & Yin, G. (2021). Reinforcement learning in healthcare: A survey. *ACM Computing Surveys*, *55*(1), 1–36.

Yungaicela-Naula, N. M., Vargas-Rosales, C., Pérez-Díaz, J. A., & Carrera, D. F. (2022). A flexible SDN-based framework for slow-rate ddos attack mitigation by using deep reinforcement learning. *Journal of Network and Computer Applications*, [ISSN: 1084-8045] *205*, Article 103444. http://dx.doi.org/10.1016/j.jnca.2022.103444.

Zaw, H. T., & Maw, A. (2019). Traffic management with elephant flow detection in software defined networks (SDN). *International Journal of Electrical and Computer Engineering*, *9*(4), 3203.

Zhang, Y., Qiu, L., Xu, Y., Wang, X., Wang, S., Paul, A., et al. (2023). Multi-path routing algorithm based on deep reinforcement learning for SDN. *Applied Sciences*, [ISSN: 2076-3417] *13*(22), http://dx.doi.org/10.3390/app132212520.

Zhang, L., Zhu, T., Hussain, F. K., Ye, D., & Zhou, W. (2023). A game-theoretic method for defending against advanced persistent threats in cyber systems. *IEEE Transactions on Information Forensics and Security*, *18*, 1349–1364. http://dx.doi.org/10.1109/TIFS.2022.3229595.

Zheng, Y., Fu, H., Xie, X., Ma, W.-Y., & Li, Q. (2011). Geolife GPS trajectory dataset - user guide. Geolife GPS trajectories 1.1 ed., URL: https://www.microsoft.com/en-us/research/publication/geolife-gps-trajectory-dataset-user-guide/.

Zolotukhin, M., Kumar, S., & Hämäläinen, T. (2020). Reinforcement learning for attack mitigation in SDN-enabled networks. In *2020 6th IEEE conference on network softwarization* (pp. 282–286). http://dx.doi.org/10.1109/NetSoft48620.2020.9165383.