electronics

MDPI

Article

Dynamic Cyberattack Simulation: Integrating Improved Deep Reinforcement Learning with the MITRE-ATT&CK Framework

Sang Ho Oh [1], Jeongyoon Kim [2] and Jongyoul Park [2,*]

[1] Department of Computer Engineering and Artificial Intelligence, Pukyong National University, Busan 48513, Republic of Korea; shoh0320@pknu.ac.kr
[2] Department of Applied Artificial Intelligence, Seoul National University of Science and Technology, Seoul 01811, Republic of Korea; kjy97426@seoultech.ac.kr
* Correspondence: jongyoul@seoultech.ac.kr; Tel.: +82-2-970-9776

Abstract: As cyberattacks become increasingly sophisticated and frequent, it is crucial to develop robust cybersecurity measures that can withstand adversarial attacks. Adversarial simulation is an effective technique for evaluating the security of systems against various types of cyber threats. However, traditional adversarial simulation methods may not capture the complexity and unpredictability of real-world cyberattacks. In this paper, we propose the improved deep reinforcement learning (DRL) algorithm to enhance adversarial attack simulation for cybersecurity with real-world scenarios from MITRE-ATT&CK. We first describe the challenges of traditional adversarial simulation and the potential benefits of using DRL. We then present an improved DRL-based simulation framework that can realistically simulate complex and dynamic cyberattacks. We evaluate the proposed DRL framework using a cyberattack scenario and demonstrate its effectiveness by comparing it with existing DRL algorithms. Overall, our results suggest that DRL has significant potential for enhancing adversarial simulation for cybersecurity in real-world environments. This paper contributes to developing more robust and effective cybersecurity measures that can adapt to the evolving threat landscape of the digital world.

Keywords: adversarial simulation; computer network; cybersecurity; deep reinforcement learning; real-world scenario

Citation: Oh, S.H.; Kim, J.; Park, J. Dynamic Cyberattack Simulation: Integrating Improved Deep Reinforcement Learning with the MITRE-ATT&CK Framework. *Electronics* **2024**, *13*, 2831. https://doi.org/10.3390/electronics13142831

Academic Editor: Andreas Mauthe

Received: 27 June 2024
Revised: 12 July 2024
Accepted: 16 July 2024
Published: 18 July 2024

1. Introduction

Over the past few years, there has been an increasing incorporation of machine learning (ML) methods within the cybersecurity domain to combat the continuously advancing complexity of cyber threats [1]. These ML algorithms have been applied to improve multiple facets of cybersecurity, such as identifying anomalies, detecting threats, responding to incidents, assessing risks, and detecting attacks on cyber–physical systems [2–9]. These advancements in ML have brought about significant changes in how organizations approach cybersecurity, allowing for a more proactive and effective defense framework against cyber threats. However, along with the benefits, there are also challenges and concerns related to the use of ML in cybersecurity, such as issues of data privacy, bias, and adversarial attacks. Therefore, it is necessary to develop techniques to deal with the current situation of cybersecurity in the context of ML, including its potential advantages and limitations, to effectively mitigate cyber threats in today's technology-driven world.

The landscape of cyberattacks has evolved, with attackers becoming more knowledgeable and resourceful in selecting and executing sophisticated techniques [10,11]. This has made traditional defense mechanisms less effective in detecting these multifaceted attacks [12,13]. As a result, protecting network security has become more challenging as cyber threats continue to advance in complexity. Traditional data mining techniques such as ML, deep learning, and statistical methods struggle to adapt to the evolving threat landscape. To address these changes, a new decision-making perspective is needed, where

security experts make decisions based on specific threat scenarios, which requires a targeted and nuanced approach. This underscores the necessity for ongoing research and innovation in cybersecurity to stay ahead of the constantly evolving nature of cyberattacks.

Adversarial simulation using ML is a growing technique in the field of cybersecurity, where ML models are trained to recognize patterns of behavior indicative of cyberattacks. By simulating attacks on a system using these ML models, organizations can gain a better understanding of the potential impact of cyberattacks and improve their overall security posture. Real-time detection and response to simulated attacks enable ML algorithms to provide proactive defense against cyber threats. The rapid advancements in artificial intelligence (AI) have opened up possibilities for AI-assisted, or even autonomous, red teaming in cybersecurity. Through AI learning and training, superior dynamic decision-making capabilities can be developed, enabling emerging attack actions across complex networked cyber systems that may not be anticipated or developed by human red team experts [14]. These red teams leverage AI to develop new and innovative ways to attack complex cyber systems, often identifying vulnerabilities that human red team experts may not have discovered yet [15]. This allows organizations to stay ahead of cyber attackers and proactively protect their systems against potential threats. Utilizing ML for adversarial simulation has proven to be a highly effective strategy in bolstering cybersecurity defenses. As cyber threats become more sophisticated, the integration of ML techniques is anticipated to become even more crucial, enabling organizations to better identify and mitigate potential risks.

Recently, reinforcement learning (RL) has attracted significant interest in cybersecurity due to its distinctive capability to create algorithms that can detect and counteract previously unknown attacks [16–18]. RL, a subset of ML, involves an agent learning optimal actions by interacting with its environment to maximize rewards [19]. Unlike traditional supervised learning that relies on labeled data, RL learns from trial and error, making it well-suited for cybersecurity where labeled datasets may not always be available. RL is highly adaptable to new and evolving threats as it constantly explores the environment and learns from experiences, enabling the development of robust and effective strategies for cyberattack detection and response. Furthermore, RL can help mitigate the risks of false positives and false negatives, which are challenges in security systems, by optimizing the detection process and reducing both unnecessary alerts and missed detections [20]. Implementing RL in cybersecurity can significantly improve the precision and efficiency of threat detection and response, thereby serving as a valuable asset in combating cyberattacks. The availability of open-source environments, such as OpenAIGym [21] and AI Safety Gridworlds [22], has significantly advanced research RL by offering standardized platforms for developing and benchmarking RL algorithms across various application domains.

Although RL-based methods have the potential to significantly improve cybersecurity [23], several challenges must be overcome. These include the limited availability of training data and the difficulties associated with modeling dynamic and complex attack scenarios [24–27]. The complexity of cyber networks has made it challenging to develop RL/deep RL (DRL) training environments for networked cyber operations. Currently, there are few RL/DRL training environments available for networked cyber operations and they are mostly in the form of simulations [28–30]. The simulation developed by Microsoft that is open sourcing its network cyber-RL training environment is called "CyberBattleSim", which is built on the OpenAIGym framework [30]. CyberBattleSim is designed to train red agents by simulating the lateral movement stage of a cyberattack within an environment that mimics a fixed network with pre-configured vulnerabilities. This simulation-based approach is reasonable due to the benefits of using abstracted models of cyber networks and operations, such as hardware resource efficiency, no real networks or action executions, and faster training. However, a potential drawback of simulated environments is their lack of realism, which can result in the trained agent's decision model being less effective in real-world scenarios. Overcoming these challenges and developing more realistic and comprehensive training environments for RL/DRL in cybersecurity could lead to

significant advancements in the field and contribute to improved cybersecurity strategies and defenses.

In a recent preliminary study on red team simulation, Sultana et al. showed that both Deep-Q-Network (DQN) and Proximal Policy Optimization (PPO) agents were able to be trained to stage and execute optimized network attack sequences, even in the presence of uncertain success distributions associated with their attack actions [31]. This demonstrates the potential of utilizing AI, specifically RL, and DRL, in enhancing the capabilities of red teaming for cyberattack simulations, allowing for more effective testing and evaluation of cybersecurity defenses against advanced and unforeseen threats. Applebaum et al. analyze RL-trained autonomous agents through a series of experiments that looked at various network settings [32]. Similarly, Elderman et al. concentrated on simulations of cybersecurity in networks that are treated as Markov games with stochastic elements and partial information [33]. They demonstrated the final game, which is a sequential adversarial decision-making issue involving two agents—the attacker and the defense.

Traditional DRL algorithms often face several critical challenges, such as slow convergence rates, high computational costs, and difficulty in handling continuous action spaces effectively. These issues can lead to inefficient learning processes and suboptimal performance, particularly in complex environments like cyberattack scenarios. In this work, we propose a rapid actor–critic (RAC) algorithm that addresses these limitations by introducing a two-step Temporal-Difference (TD) error calculation, which enhances the accuracy and speed of learning. The main outcomes of our improvements include faster convergence, reduced computational overhead, and improved efficacy in detecting and mitigating cyberattacks. By leveraging these enhancements, the RAC algorithm demonstrates superior performance compared to traditional DRL methods, as evidenced by our experimental results.

The study makes four significant contributions. First of all, it uses DRL in adversarial attack simulations successfully. Significant learning curves of the agent can be shown by modeling and simulating potential hacks and their effects on a system. This demonstrates the promise of DRL algorithms in cybersecurity by facilitating the development of automated defensive systems that are adaptive and able to react instantly to changing cyber threats. Secondly, the study contributes to the establishment of a realistic and scalable experimentation environment for DRL-based approaches in cybersecurity by utilizing an actual cyber-threat scenario from MITRE ATT&CK [34] and developing a simulation framework for it. Thirdly, we provide a better DRL algorithm called RAC to enhance red team agent performance. Finally, we made a comparison between our proposed method with the standard Q-learning algorithm, as well as additional DRL algorithms including DQN [35], actor–critic [36], and PPO [37]. This comparison showed the agents' learning curves and emphasized how important it is to choose the suitable algorithm for the explicit simulation scenario. All things considered, the study demonstrates how DRL-based techniques can improve cybersecurity by producing the best outcomes for the modeling of aggressive cyberattacks.

The structure of our paper is as follows: Section 2 describes the methodology which includes the RAC algorithm, simulation settings, and process in detail. The experimental results, including the reward obtained, agent attack success rates, and the number of iterations per episode, are presented in Section 3. We discuss the results in Section 4. Our findings and conclusions are presented in Section 5. Section 6 concludes by outlining the study's limitations and potential directions for further research.

2. Materials and Methods

RL approaches develop methods that maximize predicted returns over predetermined periods in an attempt to solve Markov decision processes (MDPs) [38]. Similar to MDPs, RL uses state, action, and reward to interact with the environment and allow the agent to make decisions based on past performance and consequences as well as the environment's

current state. RL approaches are applied in many domains, including cybersecurity, where thorough search procedures might not always find the best answers.

In this research, we employed a simulated setting that enabled our agent to interact with an authentic cyberattack situation grounded in MITRE ATT&CK. Through accelerated policy learning, this approach allowed the agent to learn from mistakes and modify its tactics in real time to counter evolving cyber threats. Furthermore, the simulated environment provided a controlled testing environment that allowed us to assess the agent's performance in different scenarios. All things considered, our study shows that DRL settings are a useful tool for agent training. Through the use of a real-world cyberattack scenario from MITRE ATT&CK, we were able to evaluate the agent's effectiveness.

2.1. MITRE ATT&CK Scenario

We used a simulated environment in our study to allow our agent to interact with a real-world cyberattack scenario that was taken from MITRE ATT&CK. The agent's learning process was accelerated in this simulated environment, allowing it to gain experience and dynamically modify its tactics in response to changing cyber threats. In addition, we evaluated the agent's performance in a variety of circumstances using a controlled testing environment. All things considered, our findings highlight how useful it is to use DRL environments to train agents. In addition, by including an actual cyberattack scenario taken from MITRE ATT&CK, we assessed the agent's performance in a pertinent and demanding environment.

The MITRE ATT&CK framework serves as a thorough framework and knowledge base that improves understanding of the operational processes, tactics, and strategies used by cyber adversaries along the cyber death chain [39,40]. ATT&CK, a worldwide available compilation of enemy tactics and strategies, was created by the federally sponsored nonprofit MITRE Corporation, which specializes in government research and development. It is based on actual findings. ATT&CK creates a common vocabulary for cybersecurity professionals to communicate threat intelligence and work together on defense measures. It is updated and maintained by cybersecurity specialists from government, industry, and academia. ATT&CK helps organizations strengthen their defenses against cyberattacks and strengthen their entire cybersecurity posture by providing a thorough understanding of adversary behavior. For cybersecurity experts around the globe, ATT&CK is an essential resource that shapes the development of security solutions and the integration of threat intelligence into security operations [41,42].

The study employed a simulated scenario based on a realistic cyber threat experienced by the Ajax security team, reproducing the techniques observed during the attack. These tactics and techniques were translated into actionable steps, and the simulation framework was constructed using process acquisition methodologies.

2.2. Deep Reinforcement Learning Application

DRL combines the principles of RL with deep learning to enable agents to learn optimal behaviors in complex environments. In DRL, an agent interacts with an environment and learns to perform tasks by receiving feedback in the form of rewards or penalties.

DRL algorithms, such as DQN, PPO, and actor–critic methods, use neural networks to approximate the policy and value functions. These algorithms enable the agent to learn and generalize from high-dimensional inputs, which makes DRL suitable for applications in various fields, including robotics, gaming, finance, and cybersecurity.

In the context of our study, DRL is employed to detect cyberattacks by training an agent to identify and respond to various attack scenarios. The agent interacts with a simulated network environment, learning to recognize vulnerabilities and take appropriate actions to mitigate threats. Through continuous interaction and feedback, the DRL agent improves its ability to detect and prevent cyberattacks, providing a robust solution for network security.

2.3. Rapid Actor–Critic (RAC)

The concept of actor–critic techniques was first introduced in [43] and has been the focus of several studies since then [36]. Actor–critic methods belong to the category of policy gradient techniques, employing a differentiable parameterization to define the policy. Gradient updates are then applied to adjust these parameters, aiming to maximize returns within a localized context. The critic assesses the actor's performance and serves as the value function, facilitating the estimation of gradients applied to the actor's updates. This reliance on gradient-based policy updates renders actor–critic techniques well suited for continuous action spaces [44].

The RAC algorithm is designed to optimize the detection and mitigation of cyberattacks in a network environment by leveraging an actor–critic architecture, as shown in Figure 1. The overall process of the RAC algorithm can be described in several key steps:

A. Initialization: The algorithm initializes with a set of parameters, including the policy parameters (θ), value function parameters (ϕ), and a replay buffer to store experiences. The environment is also initialized with the client PC and target PCs, each configured with specific security settings.

B. State Representation: The state (s) in the RAC algorithm includes all relevant information about the current situation in the network environment, such as the number of ports blocked by firewalls, administrator privileges, registry of authorized PCs, and known vulnerabilities.

C. Action Selection: The agent selects an action (a) based on the current state and the policy (π_θ). The policy is a differentiable parameterization that guides the agent in choosing actions that maximize expected returns, as shown in Equation (1).

$$a_t \sim \pi_\theta(a_t|s_t) \tag{1}$$

where a_t is the action selected at time t given the state s_t.

D. Environment Interaction and Reward Collection: The agent interacts with the environment by taking the selected action, resulting in a new state (s_{t+1}) and a reward (r_{t+1}). The reward function is designed to provide positive feedback for successful actions and negative feedback for unsuccessful actions, as shown in Equation (2).

$$s_{t+1}, r_{t+1} = \mathrm{Env}(s_t, a_t) \tag{2}$$

E. TD Error Calculation: The TD error (δ) is calculated using the RAC-specific TD error function. This function takes into account the immediate reward, the discounted future rewards, and the estimated value of future states. In order to lower bias and enable selection of more accurate attacks with faster convergence, we designed a 2-step TD actor–critic architecture. The TD error is given in Equation (3).

$$\delta_t = r_{t+1} + \gamma r_{t+1} + \gamma^2 V_\phi(s_{t+2}) - V_\phi(s_t) \tag{3}$$

where r_{t+1} is the immediate reward, γ is the discount factor, and $V_\phi(s_{t+2})$ is the value function at state s_{t+2} parameterized by ϕ.

F. Policy and Value Function Updates: Using the calculated TD error, the policy parameters (θ) and the value function parameters (ϕ) are updated. The policy gradient update is given in Equation (4).

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta log \pi_\theta(s_t, a_t) \times \delta_t] \tag{4}$$

The value function update is performed by minimizing the mean squared error of the TD error, as shown in Equation (5).

$$\phi \leftarrow \phi - \alpha \nabla_\phi \left(\delta_t^2\right) \tag{5}$$

where α is the learning rate.

The agent iteratively interacts with the environment, continuously collecting experiences, calculating TD errors, and updating the policy and value function until the training converges. This iterative process allows the agent to learn effective strategies for detecting and mitigating cyberattacks. By following these steps, the RAC algorithm ensures rapid and accurate convergence, enabling the agent to select optimal actions in the context of network security.

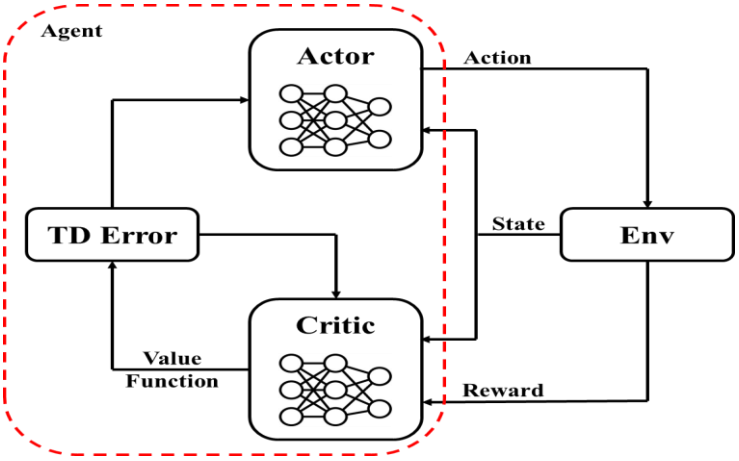

Figure 1. Structure of rapid actor–critic algorithm.

2.4. Employment of Deep Reinforcement Learning in Simulation

The PyTorch (Version 1.7.1) framework was employed to develop the simulated cyberattack scenario in this study. The researchers defined categories for the personal computer (PC), attacker, and various settings within the framework, enabling them to create and execute the simulation effectively. As shown in Figure 2, the simulated scenario comprised a client PC and four target PCs. After obtaining attack data from the client computer, the attacker launched a string of attacks meant to compromise the target computers. When the attacker successfully takes over every PC in the network environment, the simulation comes to an end.

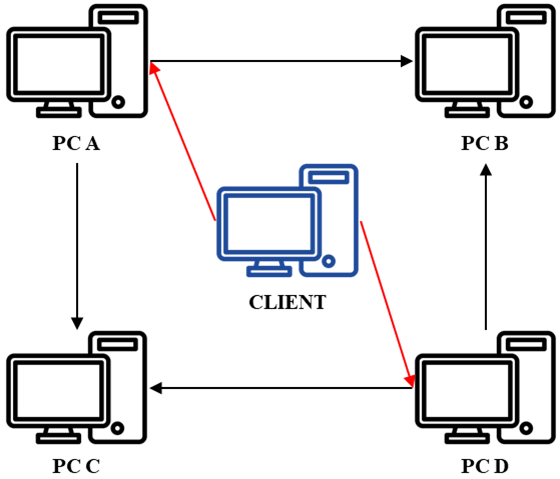

Figure 2. Simulated network environment.

The DRL parameters are employed within the attack scenario as follows: the environment initializes with a client PC and four target PCs and the agent starts by exploring the environment to identify target nodes. The agent uses the state information to decide which action to take next, based on which, it either successfully exploits a vulnerability

(gaining a positive reward) or fails (incurring a negative reward). The agent updates its policy and value function based on the rewards received from its actions, continuously learning from its actions to improve its strategy for future scenarios. The agent simulates attacks on identified target nodes, adjusting its actions based on the state and received rewards to maximize its success rate.

2.4.1. States

States, actions, and rewards compose MDPs. In this context, the state denotes the state of the individual PC within the network. Table 1 lists the four characteristics that this study found affect a PC's vulnerability to cyberattacks. The elements of the state are as follows: (1) the number of ports blocked by firewalls; (2) administrator privileges granted by user credentials; (3) a registry of authorized PCs that can share administrator privileges without credentials; and (4) vulnerabilities that may expose the credentials of other PCs linked to the user's computer. Without keyboard protection, keylogging becomes feasible and web browsers may retain credentials. The danger of contracting a virus rises when you open an unsecured email. To make informed decisions in DRL, agents need to have a thorough awareness of the condition of the environment. This state is made up of a collection of variables that characterize the surroundings about the goals of the agent.

Table 1. State components.

No.	State
1	The Number of Ports Blocked by Firewalls
2	Administrator Privileges Granted by User Credentials
3	The Roster of Authorized PCs
4	Keyboard Security

2.4.2. Actions

Table 2 shows four possible actions for the attacker, which include (1) conducting key logging to acquire credentials in the absence of keyboard security, (2) accessing web-stored credentials, (3) attempting to access ports through open ports, and (4) impersonating an authorized PC's IP address for login. After deciding which exploit technique to use to target the PC's weaknesses, the attacker performs a port scan to find open ports that could be exploited. The number of linked nodes, previously discovered open ports from earlier attacks, the presence of keyboard security features, and the presence of web credentials are the four attributes that the attacker uses to describe the state. The attacker chooses the best course of action to take against the target PC based on the knowledge about its present status.

Table 2. Action components.

No.	Action
1	Key Logging
2	Credential from Web Browser
3	Opened Port Attack
4	Spoofing

Several factors, including user credential selection or vulnerability, must be taken into account to carry out efficient activities in DRL-based cyberattack simulations. Every action has prerequisites that must be fulfilled, like finding the target host and having the necessary credentials. These actions could result in the identification of new hosts, the gathering of personal data, or the compromise of other hosts.

2.4.3. Rewards

The reward function is used in a DRL cyberattack simulation to give the agent feedback on how effective its actions were. This feature takes into account the cost of actions as well as how they affect the penetration testing procedure. To be more specific, the agent determines the reward value of each action it takes by considering both its cost and effectiveness. In this study, if the agent can effectively take control of the node, a positive incentive (+1) is given; if not, a negative reward (−1) is administered. The agent can learn from this feedback and modify its behavior to obtain the best outcomes in changing circumstances.

2.5. Simulation Framework

2.5.1. Data Gathering Process

The training data for the DRL agent is gathered through interactions with a simulated network environment. The environment consists of a client PC and multiple target PCs with varying security levels that is designed to simulate real-world scenarios. The DRL agent interacts with this environment to gather experiences, which are used for training. The process of gathering training data involves several key steps: (1) environment initialization; (2) state representation; (3) agent action selection; and (4) interaction and reward collection.

By following these steps in simulation, the DRL agent gathers comprehensive training data that encompasses a wide range of attack scenarios and security configurations. These data are crucial for training the agent to detect and respond to cyberattacks effectively.

2.5.2. Simulation Process

The simulation process involves two phases: discovering nodes and attacking nodes. In the node discovery phase, illustrated in Figure 3, the attacker identifies targets for the attack. If the attacker successfully exploits a vulnerability, they can obtain the credentials required to access the discovered nodes. However, if they fail to exploit the vulnerability, they can only discover the connected nodes without obtaining any credentials. After completing this process, the attacker can use the acquired credentials to target the identified nodes. This step is essential to locate additional nodes and gather the required credentials within the network.

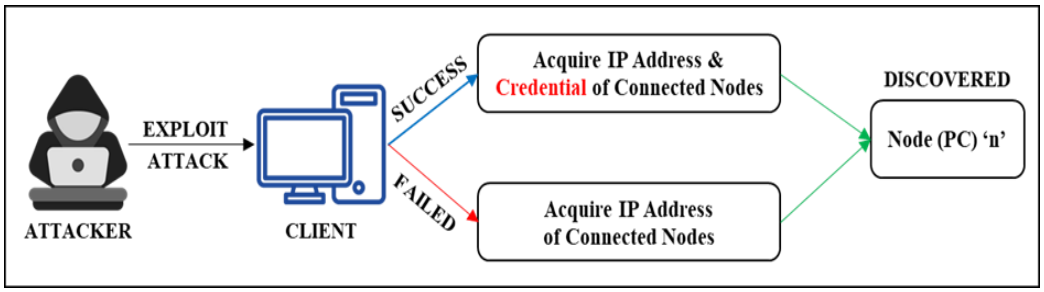

Figure 3. Process of discovering nodes.

In the second part of the simulation, the attacker attempts to compromise the nodes discovered in the previous phase. The attacker can take direct control of these nodes if they have the necessary credentials. However, if they do not have the credentials, the attacker first employs spearphishing attacks to target the nodes. Once the attacker has established whether the targeted nodes are compromised, they will use one of four attack techniques: spoofing, open port attacks, key logging, or credential theft from web browsers. Until they manage to obtain the node's credentials, the attacker switches back and forth between these methods. Once the attacker has the credentials, they can take control of the node and use it to launch further attacks on new nodes in the network. Figure 4 illustrates this process.

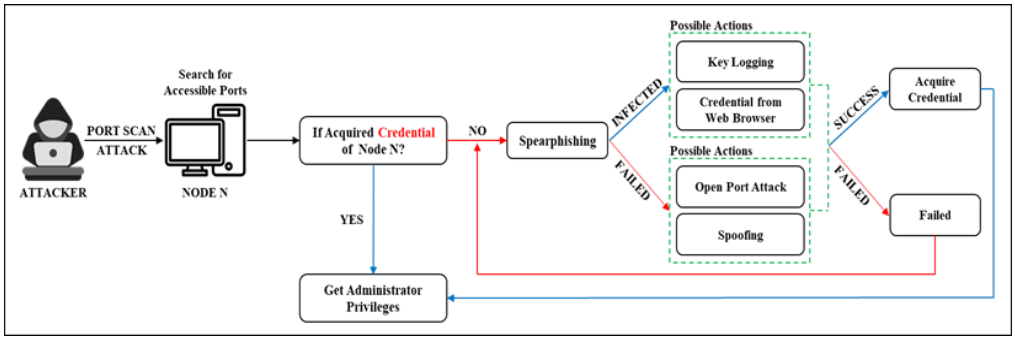

Figure 4. Process of attacking a node.

It is crucial to point out that the attack strategies employed in this simulation are based on actual cyberattack strategies and tactics, giving a realistic picture of the risks that computer networks confront. It is feasible to develop and assess successful defense tactics against these attacks by mimicking them in a controlled environment. The two-stage simulation process's integration of node attack and discovery provides a thorough and accurate environment for evaluating how well DRL agents respond to cyberattack scenarios.

In the cyberattack simulation that we conducted, a PC is considered "owned" if the attacker successfully obtains administrator privileges after attacking it. Whenever the attacker succeeds in owning a PC, they receive a reward of +1, whereas if they fail, they receive a reward of −1.

3. Results

In this section, we outline the outcomes of a comparison between the RAC agents and other current DRL algorithms in terms of cumulative rewards. The results suggest that the RAC algorithms facilitated the agent in developing a more effective attack strategy. The simulation results demonstrate that the RAC agents' actions were successful in entering a simulated cyber environment and acquiring successful methods for attack. The graphs presented in this section illustrate the rewards earned by the agents across episodes, their attack success rate, and the iterations took per episode, demonstrating an improvement in the agents' ability to achieve higher rewards over the course of each episode.

To showcase the effectiveness of RAC in cybersecurity tasks, the study compared the performance of RAC algorithms with four other RL algorithms: DQN, actor–critic, PPO, and Q-learning.

3.1. Comparison of Reward Obtained by RL Algorithms

Together with other DRL algorithms, the average reward obtained by RAC in the experimental network setting is shown in Figure 5. When the attacker chooses actions at random to compromise target nodes, the algorithm initially displays a low mean reward and a low success rate. As the training progresses, the attacker gradually learns the optimal policy, which involves selecting the action with the highest success probability. As a consequence, the DRL algorithm's mean reward increases, indicating that the attacker's success rate in taking over the target node has also increased. The graph shows that there are fluctuations in the mean reward attained by the algorithm. This is because the algorithm might become trapped in a local minimum, where it is unable to discover the optimal policy. However, with continued training, the algorithm can ultimately overcome the local minimum and continue to enhance its performance.

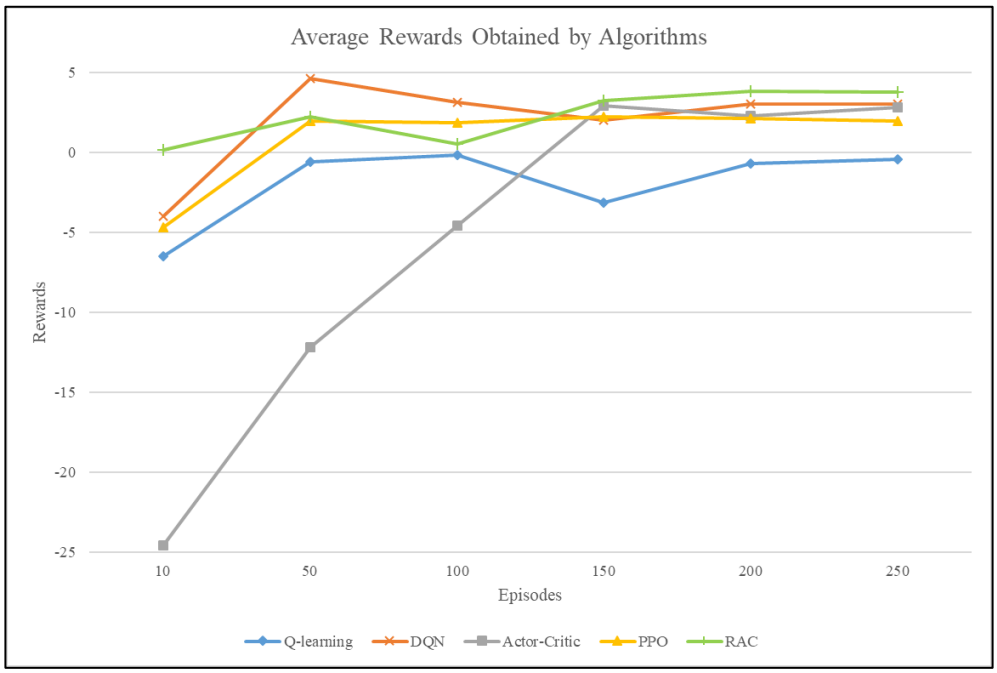

Figure 5. Average rewards obtained by RL agents.

The results show that because the tabular Q-learning algorithm uses dynamic programming, which stores data in a table rather than a neural network, it performs better at first than actor–critic algorithms. Q-learning, however, performs less well than the other algorithms as training proceeds through 100 and 150 episodes and the volume of data grows. When the state and action space increase, it requires long training times and large memory, which makes Q-learning increasingly unstable. The DQN algorithm, despite using soft updates, becomes stuck in local minima after more than 150 episodes. If it becomes trapped in a local minimum, its performance will remain constant even if training continues unless it escapes. As opposed to value-based algorithms like DQN and Q-learning, the suggested RAC algorithm is policy-based and performs well in the experiment, allowing for more flexible decision making.

As shown in Table 3, at the initial episode, the average reward obtained by Q-learning is −6.5, while actor–critic's reward is −24.55. However, as the training progresses, at 150 episodes the RL algorithms with neural networks have better performance. At the end of the training, our proposed RAC algorithm showed the highest average reward of 3.8. This section presents the reward obtained by five RL algorithms including our proposed RAC algorithm. The average rewards obtained by the agents for each episode are shown, and it can be observed that the agents learn to obtain a higher reward as the episode progresses. The outcomes demonstrate how the DRL approaches enable agents to carry out effective attack strategies while emphasizing the effectiveness of the suggested RAC algorithms in the cybersecurity domain.

Table 3. Average reward obtained by RL algorithms.

Algorithms	**Episodes**				
	50	**100**	**150**	**200**	**250**
Q-learning	−6.5	−0.6	−0.2	−3.15	−0.7
DQN	−4.0	4.6	3.1	2.0	3.0
Actor–Critic	−24.55	−12.2	−4.6	2.9	2.25
PPO	−4.7	1.95	1.85	2.2	2.1
RAC	0.15	2.2	0.5	3.2	3.8

3.2. Comparison of the Attack Success Rates Achieved by RL Algorithms

In this experiment, the attack success rate of the proposed RAC algorithm and other DRL algorithms were evaluated in a simulation environment, as shown in Figure 5. The attack success rate was determined by whether the attacker successfully infiltrated and owned all the PCs in the simulated network environment. A similar pattern can be seen in the reward graph and success rate shown in Figure 6, where larger incentives are associated with higher success rates. The RAC algorithm's ability to successfully penetrate a simulated cyber environment is further evidenced by the graph, which displays the greatest success rate after training.

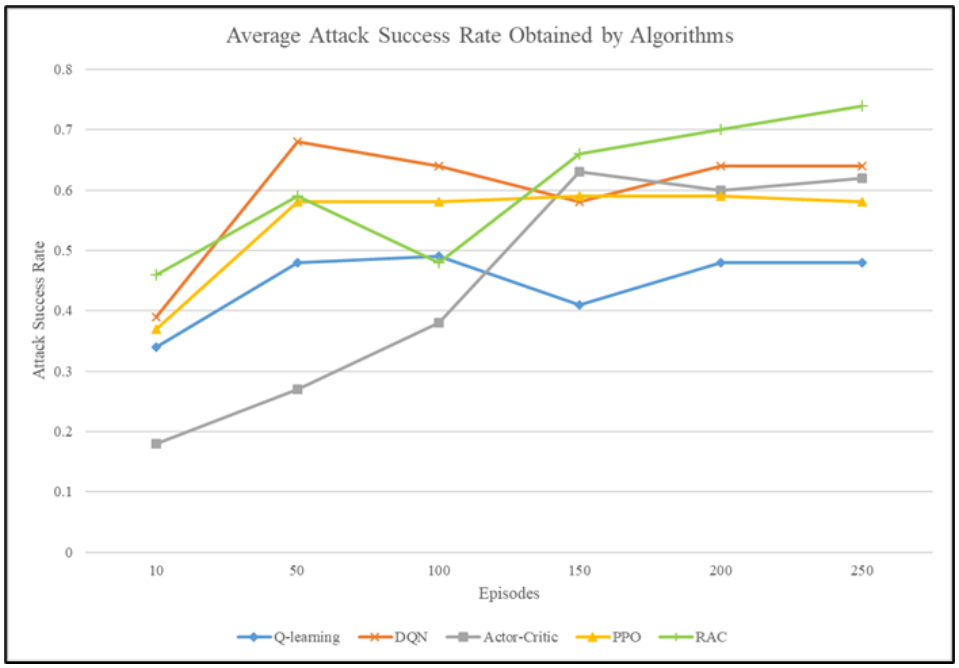

Figure 6. Average success rate obtained by RL agents.

The results show that our suggested RAC method is successful in achieving the objective of penetrating a cyber-simulation. Moreover, the success rate and reward graph trends suggest that the RAC algorithm was able to balance the exploration–exploitation tradeoff effectively, leading to higher success rates and rewards. This suggests that the algorithm is learning and improving its performance over time. Furthermore, the results highlight the importance of evaluating the success rate in evaluating the performance of DRL algorithms in cybersecurity applications. By using the success rate as a metric, researchers can determine how effective an algorithm is at achieving its objective and how it compares to other algorithms.

Our results show that at the initial stage of the training, which is at 10 episodes, RAC shows an average success rate of 0.46, while the other algorithms, Q-learning, DQN, actor–critic, and PPO, have success rates of 0.34, 0.39, 0.18, and 0.37, respectively, as shown in Table 4. These findings indicate that the RAC algorithm has better initial learning speed and can achieve higher performance than other RL algorithms in the early stages of the training. This is an essential discovery because it demonstrates how quickly the RAC algorithm can learn from and adjust to a new environment in comparison to other RL algorithms.

Table 4. Average success rate obtained by RL algorithms.

Algorithms	Episodes				
	50	100	150	200	250
Q-learning	0.34	0.48	0.49	0.41	0.48
DQN	0.39	0.68	0.64	0.58	0.64
Actor–Critic	0.18	0.27	0.38	0.63	0.6
PPO	0.37	0.58	0.58	0.59	0.59
RAC	0.46	0.59	0.48	0.66	0.74

Furthermore, the results show that at the end of the training, RAC achieved the highest average success rate of 0.74. This indicates that the RAC algorithm is effective in learning and improving its performance over time. It also demonstrates that the proposed algorithm outperforms the other DRL algorithms in the given task. However, it is crucial to note that the success rate is not the only metric that should be considered when evaluating the performance of RL algorithms in a given task. It is important to evaluate the success rate and the rewards to have a broad understanding of the performance. Overall, the results suggest that the proposed RAC algorithm shows promising results in learning and improving its performance over time compared to other DRL algorithms.

3.3. Comparison of the Number of Iterations for Each RL Algorithm

The experimental results presented in this section shed light on the learning efficiency of our proposed RAC and other RL algorithms in a simulation environment by measuring the count of iterations required to execute each episode. An iteration refers to the attempted action to complete one episode. As depicted in Figure 7, out of all the investigated algorithms, the proposed RAC algorithm demonstrates the most effective learning, needing the fewest iterations to finish each episode. This suggests that, in comparison to other algorithms, the RAC algorithm more quickly and efficiently traverses complicated state and action domains. In contrast, the Q-learning algorithm shows limited improvement in learning efficiency, as indicated by the consistent number of iterations throughout the learning process. This could stem from the exploration–exploitation dilemma, where the agent faces challenges in balancing between exploring new actions and exploiting existing knowledge. The Q-learning algorithm may be too reliant on its current knowledge, resulting in suboptimal learning efficiency.

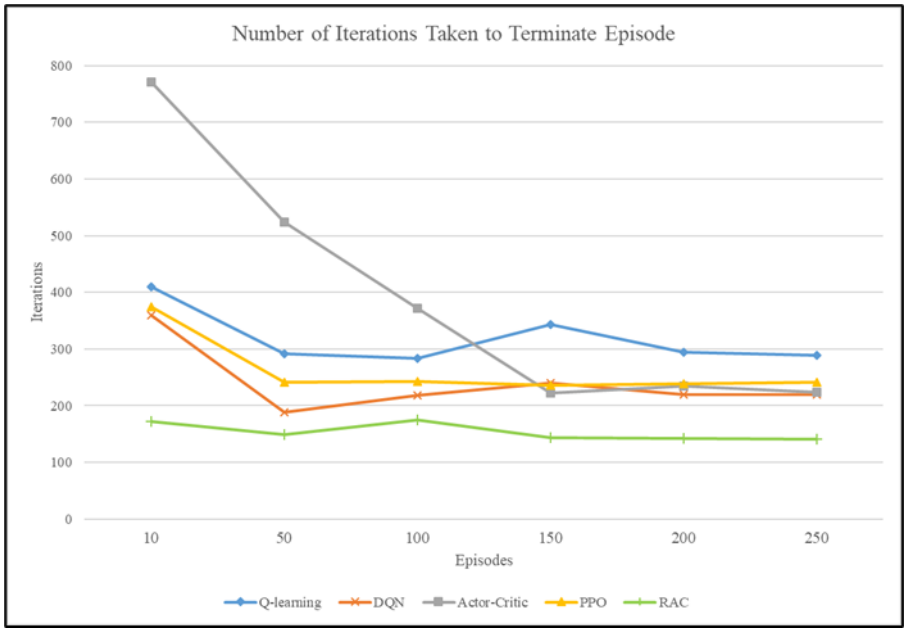

Figure 7. Number of iterations taken by RL algorithms.

This section's experimental findings contrast our proposed RAC algorithm's learning efficiency with that of other DRL algorithms. According to Table 5, the RAC algorithm demonstrates superior efficiency, requiring only 141 iterations to complete an episode. In contrast, the Q-learning, DQN, actor–critic, and PPO algorithms require more iterations, 289, 220, 224, and 241, respectively, to achieve the same level of success. This indicates that the RAC algorithm is able to learn faster and more efficiently compared to other algorithms, which is critical in real-world applications where time is of the essence.

Table 5. Average number of iterations for each RL algorithm.

Algorithms	Episodes					
	20	**50**	**100**	**150**	**200**	**250**
Q-learning	410	292	284	343	294	289
DQN	360	188	218	240	220	220
Actor–Critic	771	524	372	222	235	224
PPO	374	241	243	236	238	241
RAC	172	149	175	144	142	141

The results of this study also provide insightful information about how different algorithms function during the learning process, which helps choose the best algorithm for a given task. For example, the DQN algorithm's reliance on value-based methods results in a tendency to select actions based solely on their highest cumulative reward value, potentially restricting its capacity to explore diverse strategies. These limitations of the DQN algorithm can be addressed by using more sophisticated techniques such as the RAC algorithm proposed in this study.

4. Discussion

To assess our proposed DRL algorithms, we successfully created and implemented a real-world cyberattack scenario in this study. This scenario simulated a realistic network environment, incorporating various vulnerabilities and obstacles typically encountered by attackers. Our research demonstrates that employing DRL in adversarial simulations has the potential to enhance cybersecurity measures by offering a more precise assessment of system vulnerabilities. This approach enables organizations to evaluate their cybersecurity posture more efficiently and cost effectively. We proposed an improved DRL algorithm called RAC to boost the performance of the red team agent. The findings demonstrated that, in terms of learning rate and cumulative reward, our proposed framework performed better than the state-of-the-art RL algorithms. These outcomes demonstrate how well our proposed algorithm generates a variety of plausible attacks that can evade current security protocols.

It is clear from the findings that the proposed RAC algorithm performs better in terms of cumulative rewards than the other DRL algorithms. The RAC algorithm facilitated the agent in developing a more effective attack strategy, allowing the agent to enter a simulated cyber environment and acquire successful methods for attack. The results of the study indicate that the proposed RAC algorithm can increase the performance in terms of learning rate and cumulative reward, which is essential for evaluating cybersecurity posture. The findings suggest that traditional Q-learning algorithms can perform well at the beginning of the training process by obtaining a higher average reward than actor–critic. However, as the training process progresses and the amount of data grows, the RL algorithms with neural networks have better performance. This is because Q-learning requires long training times and a large memory, making it increasingly unstable. Similarly, the DQN algorithm, despite using soft update, becomes stuck in local minima after more than 150 episodes, leading to constant performance unless it escapes. In contrast to value-based algorithms like Q-learning and DQN, the proposed RAC algorithm, which is distinguished by its policy-based methodology, demonstrated superior results in the experiment by permitting more flexible decision making. By the end of the training, the RAC algorithm demonstrated the

highest average reward, highlighting its effectiveness in enhancing adversarial simulations for cybersecurity in real-world environments.

The experiment also evaluated the success rate of the RAC algorithm and other DRL algorithms in a simulation environment. The success rate was measured by whether the attacker could infiltrate and own all the PCs in the simulated network. As shown in Table 4, the results indicate that the RAC algorithm had the greatest average success rate at the initial training stage, at 0.46. Other RL algorithms that had success rates at this level were Q-learning, DQN, actor–critic, and PPO, with 0.34, 0.39, 0.18, and 0.37, respectively. This finding implies that the RAC algorithm has a better initial learning speed and can achieve better performance than other RL algorithms in the early stages of training. This observation is significant because it indicates that the RAC algorithm can learn and adapt to a new environment faster than other RL algorithms. Moreover, the RAC algorithm achieved the greatest average success rate of 0.74 after the training. These findings show that by deftly managing the exploration–exploitation tradeoff, the RAC algorithm accomplished the goal of penetrating the simulated cyber environment, leading to higher success rates and rewards. The results highlight how important it is to compare DRL algorithms' performance to other algorithms and evaluate their efficacy in cybersecurity applications using success rate as a criterion.

Lastly, we evaluated the learning efficiency of our proposed RAC algorithm and other RL algorithms in a simulation environment by measuring the number of iterations needed to complete each episode. An iteration refers to an attempted action to complete one episode. The RAC algorithm demonstrates the most efficient learning among the tested algorithms, requiring only 141 iterations to complete each episode. In contrast, the Q-learning, DQN, actor–critic, and PPO algorithms require more iterations, 289, 220, 224, and 241, respectively, to achieve a comparable level of success. This suggests that the RAC algorithm effectively navigates complex state and action spaces more efficiently and swiftly compared to other algorithms. However, the Q-learning algorithm shows limited improvement in learning efficiency, as evidenced by the consistent number of iterations throughout the learning process. This could be attributed to the exploration–exploitation dilemma, where the agent faces challenges in balancing between exploring new actions and exploiting existing knowledge. Additionally, the findings provide information on the limitations of different algorithms during the learning process, which helps choose the best algorithm for a given task. For example, because the DQN algorithm always selects actions based on the action with the highest cumulative reward value, its value-based design may limit its ability to investigate a variety of tactics. These limitations of the DQN algorithm can be overcome by using more advanced techniques such as the RAC algorithm proposed in this study.

5. Conclusions

To demonstrate the potential use of DRL in cybersecurity, we implemented a realistic cyberattack scenario and presented an improved DRL algorithm, called RAC, in this paper. The performance of the RAC algorithm was compared to that of other cutting-edge RL algorithms, such as Q-learning, DQN, actor–critic, and PPO, in the study.

The study underscores the importance of evaluating metrics such as success rate, reward, and learning efficiency to gain a comprehensive understanding of an algorithm's performance. Evaluating the agent's success rate and rewards is essential to determining how well an algorithm performs in accomplishing its goals. The results of the study can also be used to choose the best algorithm for a given task and provide insightful information about how various algorithms function when learning.

According to the study's findings, the RAC algorithm performs better than other DRL algorithms in terms of earning rewards, hitting success rates, being efficient at learning, and striking a balance between exploitation and exploration. The success rate results show that the RAC algorithm was most successful at breaking into a simulated cyber environment, with the highest average success rate both at the beginning and after training. Furthermore,

the learning efficiency results show that the RAC algorithm is the most efficient in terms of the number of iterations needed to finish an episode, indicating its superior speed and effectiveness in navigating intricate state and action spaces compared to other algorithms.

In conclusion, our proposed RAC algorithm has demonstrated a great deal of promise as a useful cybersecurity tool. The outcomes of the experiment highlight how DRL algorithms can be used to teach agents to successfully penetrate simulated cyber environments. These results progress the development of stronger defenses against the increasing frequency and complexity of cyberattacks. In the end, our work adds significantly to the growing body of knowledge regarding the use of DRL algorithms in cybersecurity.

6. Limitations and Future Works

DRL has a lot of potential applications in cybersecurity; however, to maximize the effectiveness of these models, some constraints and difficulties must be addressed. The dynamic nature of cyberattacks presents a constant challenge, highlighting the necessity for sophisticated simulation systems that can instantly adjust to changes in attack patterns and network configurations. Future research should prioritize resource-efficient methods like parameter sharing and model compression to meet the resource-intensive requirements of DRL training.

To enhance the applicability of DRL models from simulated to real-world environments, more research on effective transfer learning methodologies is needed. Moreover, the construction of continuous learning frameworks, interpretable DRL models, and adversarial defensive mechanisms are important areas for future research. Additionally, to guarantee that cybersecurity automated decision-making systems comply with ethical and regulatory requirements, validation and assurance frameworks must be developed. It takes teamwork among cybersecurity and ML professionals to successfully navigate and handle these challenging problems.

Author Contributions: S.H.O. and J.K. conceived and conducted the experiments. J.P. verified and guided the experimental outcomes. All authors have read and agreed to the published version of the manuscript.

Funding: This study was supported by the Research Program funded by SeoulTech (Seoul National University of Science and Technology).

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Zeadally, S.; Adi, E.; Baig, Z.; Khan, I.A. Harnessing artificial intelligence capabilities to improve cybersecurity. *IEEE Access* **2020**, *8*, 23817–23837. [CrossRef]
2. Kilincer, I.F.; Ertam, F.; Sengur, A. Machine learning methods for cyber security intrusion detection: Datasets and comparative study. *Comput. Netw.* **2021**, *188*, 107840. [CrossRef]
3. Ferrag, M.A.; Maglaras, L.; Moschoyiannis, S.; Janicke, H. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *J. Inf. Secur. Appl.* **2020**, *50*, 102419. [CrossRef]
4. Hariharan, A.; Gupta, A.; Pal, T. Camlpad: Cybersecurity autonomous machine learning platform for anomaly detection. In *Advances in Information and Communication: Proceedings of the 2020 Future of Information and Communication Conference (FICC), San Francisco, CA, CA, USA, 5–6 March 2020*; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; Volume 2, pp. 705–720.
5. Gökdemir, A.; Calhan, A. Deep learning and machine learning based anomaly detection in internet of things environments. *J. Fac. Eng. Archit. Gazi Univ.* **2022**, *37*, 1945–1956.
6. Sentuna, A.; Alsadoon, A.; Prasad, P.W.C.; Saadeh, M.; Alsadoon, O.H. A novel Enhanced Naïve Bayes Posterior Probability (ENBPP) using machine learning: Cyber threat analysis. *Neural Process. Lett.* **2021**, *53*, 177–209. [CrossRef]
7. Jakka, G.; Yathiraju, N.; Ansari, M.F. Artificial Intelligence in Terms of Spotting Malware and Delivering Cyber Risk Management. *J. Posit. Sch. Psychol.* **2022**, *6*, 6156–6165.

8. Sarker, I.H.; Furhad, M.H.; Nowrozy, R. Ai-driven cybersecurity: An overview, security intelligence modeling and research directions. *SN Comput. Sci.* **2021**, *2*, 173. [CrossRef]
9. Zhang, J.; Pan, L.; Han, Q.L.; Chen, C.; Wen, S.; Xiang, Y. Deep learning based attack detection for cyber-physical system cybersecurity: A survey. *IEEE/CAA J. Autom. Sin.* **2021**, *9*, 377–391. [CrossRef]
10. Guembe, B.; Azeta, A.; Misra, S.; Osamor, V.C.; Fernandez-Sanz, L.; Pospelova, V. The emerging threat of ai-driven cyber attacks: A Review. *Appl. Artif. Intell.* **2022**, *36*, 2037254. [CrossRef]
11. Oreyomi, M.; Jahankhani, H. Challenges and Opportunities of Autonomous Cyber Defence (ACyD) Against Cyber Attacks. In *Blockchain and Other Emerging Technologies for Digital Business Strategies*; Springer International Publishing: Berlin/Heidelberg, Germany, 2022; pp. 239–269.
12. Wazid, M.; Das, A.K.; Chamola, V.; Park, Y. Uniting cyber security and machine learning: Advantages, challenges and future research. *ICT Express* **2022**, *8*, 313–321. [CrossRef]
13. Mohammadi, F. Emerging challenges in smart grid cybersecurity enhancement: A review. *Energies* **2021**, *14*, 1380. [CrossRef]
14. Li, L.; Fayad, R.; Taylor, A. Cygil: A cyber gym for training autonomous agents over emulated network systems. *arXiv* **2021**, arXiv:2109.03331.
15. Piplai, A.; Anoruo, M.; Fasaye, K.; Joshi, A.; Finin, T.; Ridley, A. Knowledge guided Two-player Reinforcement Learning for Cyber Attacks and Defenses. In Proceedings of the International Conference on Machine Learning and Applications, Nassau, Bahamas, 12–14 December 2022.
16. Salih, A.; Zeebaree, S.T.; Ameen, S.; Alkhyyat, A.; Shukur, H.M. A survey on the role of artificial intelligence, machine learning and deep learning for cybersecurity attack detection. In Proceedings of the 2021 7th International Engineering Conference "Research & Innovation amid Global Pandemic" (IEC), Erbil, Iraq, 24–25 February 2021; pp. 61–66.
17. Nguyen, T.T.; Reddi, V.J. Deep reinforcement learning for cyber security. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *34*, 3779–3795. [CrossRef] [PubMed]
18. Balhara, S.; Gupta, N.; Alkhayyat, A.; Bharti, I.; Malik, R.Q.; Mahmood, S.N.; Abedi, F. A survey on deep reinforcement learning architectures, applications and emerging trends. *IET Commun.* **2022**. [CrossRef]
19. Moerland, T.M.; Broekens, J.; Plaat, A.; Jonker, C.M. Model-based reinforcement learning: A survey. *Found. Trends® Mach. Learn.* **2023**, *16*, 1–118. [CrossRef]
20. Caminero, G.; Lopez-Martin, M.; Carro, B. Adversarial environment reinforcement learning algorithm for intrusion detection. *Comput. Netw.* **2019**, *159*, 96–109. [CrossRef]
21. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. Openai gym. *arXiv* **2016**, arXiv:1606.01540.
22. Leike, J.; Martic, M.; Krakovna, V.; Ortega, P.A.; Everitt, T.; Lefrancq, A.; Orseau, L.; Legg, S. AI safety gridworlds. *arXiv* **2017**, arXiv:1711.09883.
23. Oh, S.H.; Jeong, M.K.; Kim, H.C.; Park, J. Applying Reinforcement Learning for Enhanced Cybersecurity against Adversarial Simulation. *Sensors* **2023**, *23*, 3000. [CrossRef]
24. Ahsan, M.; Nygard, K.E.; Gomes, R.; Chowdhury, M.M.; Rifat, N.; Connolly, J.F. Cybersecurity threats and their mitigation approaches using Machine Learning—A Review. *J. Cybersecur. Priv.* **2022**, *2*, 527–555. [CrossRef]
25. Ambalavanan, V. Cyber threats detection and mitigation using machine learning. In *Handbook of Research on Machine and Deep Learning Applications for Cyber Security*; IGI Global: Hershey, PA, USA, 2020; pp. 132–149.
26. Standen, M.; Lucas, M.; Bowman, D.; Richer, T.J.; Kim, J.; Marriott, D. Cyborg: A gym for the development of autonomous cyber agents. *arXiv* **2021**, arXiv:2108.09118.
27. Walter, E.; Ferguson-Walter, K.; Ridley, A. Incorporating deception into cyberbattlesim for autonomous defense. *arXiv* **2021**, arXiv:2108.13980.
28. Zhou, S.; Liu, J.; Hou, D.; Zhong, X.; Zhang, Y. Autonomous penetration testing based on improved deep q-network. *Appl. Sci.* **2021**, *11*, 8823. [CrossRef]
29. Baillie, C.; Standen, M.; Schwartz, J.; Docking, M.; Bowman, D.; Kim, J. Cyborg: An autonomous cyber operations research gym. *arXiv* **2020**, arXiv:2002.10667.
30. Microsoft Defender Research Team. CyberBattleSim. Created by Christian Seifert, Michael Betser, William Blum, James Bono, Kate Farris, Emily Goren, Justin Grana, Kristian Holsheimer, Brandon Marken, Joshua Neil, Nicole Nichols, Jugal Parikh, Haoran Wei. 2021. Available online: https://github.com/microsoft/cyberbattlesim (accessed on 15 January 2024).
31. Sultana, M.; Taylor, A.; Li, L. Autonomous network cyber offence strategy through deep reinforcement learning. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III*; SPIE: Bellingham, WA, USA, 2021; Volume 11746, pp. 490–502.
32. Applebaum, A.; Dennler, C.; Dwyer, P.; Moskowitz, M.; Nguyen, H.; Nichols, N.; Park, N.; Rachwalski, P.; Rau, F.; Webster, A.; et al. Bridging automated to autonomous cyber defense: Foundational analysis of tabular q-learning. In *CCS '22: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, Los Angeles, CA, USA, 7–11 November 2022*; ACM: New York, NY, USA, 2022; pp. 149–159.
33. Elderman, R.; Pater, L.J.; Thie, A.S.; Drugan, M.M.; Wiering, M.A. Adversarial Reinforcement Learning in a Cyber Security Simulation. In Proceedings of the 9th International Conference on Agents and Artificial Intelligence, Porto, Portugal, 24–26 February 2017; pp. 559–566.

34. Strom, B.E.; Applebaum, A.; Miller, D.P.; Nickels, K.C.; Pennington, A.G.; Thomas, C.B. Mitre att&ck: Design and philosophy. In *Technical Report*; The MITRE Corporation: Bedford, MA, USA, 2018.
35. Nair, A.; Srinivasan, P.; Blackwell, S.; Alcicek, C.; Fearon, R.; De Maria, A.; Panneershelvam, V.; Suleyman, M.; Beattie, C.; Petersen, S.; et al. Massively parallel methods for deep reinforcement learning. *arXiv* **2015**, arXiv:1507.04296.
36. Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. Soft actor-critic algorithms and applications. *arXiv* **2018**, arXiv:1812.05905.
37. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
38. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
39. Sutton, R.S.; McAllester, D.; Singh, S.; Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *Adv. Neural Inf. Process. Syst.* **1999**, *12*, 1057–1063.
40. The MITRE Corporation. Ajax Security Team, The MITRE Corporation. 2016. Available online: https://attack.mitre.org/groups/G0130/ (accessed on 5 December 2022).
41. Alexander, O.; Belisle, M.; Steele, J. *MITRE ATT&CK® for Industrial Control Systems: Design and Philosophy*; The MITRE Corporation: Bedford, MA, USA, 2020; pp. 1–43.
42. Strom, B.E.; Battaglia, J.A.; Kemmerer, M.S.; Kupersanin, W.; Miller, D.P.; Wampler, C.; Whitley, S.M.; Wolf, R.D. *Finding Cyber Threats with ATT&CK-Based Analytics*; Technical Report No. MTR170202; The MITRE Corporation: Bedford, MA, USA, 2017.
43. Konda, V.; Tsitsiklis, J. Actor-critic algorithms. *Adv. Neural Inf. Process. Syst.* **1999**, *12*, 1008–1014.
44. Grondman, I.; Vaandrager, M.; Busoniu, L.; Babuska, R.; Schuitema, E. Efficient model learning methods for actor–critic control. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **2011**, *42*, 591–602. [CrossRef]