A l-Noor Journal for Information Technology and Cybersecurity, Vol. 1, No.3, 2026 (64-74)

ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v3.i1.a6

Al-Noor Journal for Information Technology and Cybersecurity

https://jncs.alnoor.edu.iq/

Evaluating an Availability-Aware Reinforcement Learning–Based Defensive Policy Against Cyber Attacks

[1]Mazin Mohammed Hamid, **[2]S M Abdul Mueid**

[1] Department of Cyber Security Techniques Engineering, Technical Engineering College for Computer and Artificial Intelligence, Mosul, Ninevah, 41001, Iraq

[2]Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, Batu Pahat, 86400, Malaysia

Article information

Article history:
Received:6-3-2026
Revised:10-5-2026
Accepted:2-6-2026

Keywords:
Advanced Persistent Threats (APT)
Reinforcement Learning
Availability-Aware Defense Policy
Autonomous Cyber Defense

Correspondence:

Mazin Mohammed Hamid
Mazin.mohammed@ntu.edu.iq

Abstract

Advanced Persistent Threat (APT) attacks represent a class of highly sophisticated and targeted cyber threats that aim to achieve strategic objectives with significant impact. Defending against such attacks is inherently challenging due to their scale, persistence, and the potential exploitation of previously unknown vulnerabilities. Moreover, overly aggressive defensive actions may negatively affect system availability and disrupt normal user operations. To address this challenge, this paper proposes an availability-aware reinforcement learning–based defense policy that aims to balance effective attack mitigation with maintaining system usability. The proposed approach incorporates real-time availability indicators—specifically, the number of active processes and sessions on each asset—into the state representation of the defender agent. The policy is trained using a cyberattack simulation environment based on the cyborg platform, leveraging the Proximal Policy Optimization (PPO) algorithm. Experimental results demonstrate that incorporating availability information leads to improved overall performance. In particular, the proposed method reduces attack duration by up to 279 time steps against an optimal attacker and 31 time steps against a sequential attacker, while also decreasing the number of defensive actions that negatively impact system availability. These findings indicate that availabilityaware state representation can significantly enhance the effectiveness and practicality of reinforcement learning –based cyber defense strategies.

DOI: https://doi.org/10.69513/jncs.v3.i1.a6

1. Introduction

Cyberspace has become an active area, with an escalation in cyber attacks, in a fast digital transformation. APTs have become one of the most important threats in the cyber security arena, due to their stealthy and persistent nature and ability to target high value assets that are hard to defend by using traditional security systems.

They're well planned, targeted and persistent, generally trying to get into specific assets over an extended period of time. APTs are hard to detect and mitigate by using

ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v3.i1.a6

traditional security tools like firewalls, intrusion detection systems (IDS), and antivirus because they use a variety of advanced techniques, such as multi-stage attacks. Structured frameworks have been extensively used to understand and analyses these attacks. A framework that classifies the real world attack behaviors in terms of tactics, techniques, and procedures (TTPs)[3] is used to model and simulate realistic attack scenarios. Reinforcement learning (RL) can be employed to simulate the sequential decision making task and let a defender agent learn to optimize its actions in the environment through interaction with the environment[4]. However, in recent years, thanks to the advent of powerful computing resources and rapid advances in deep reinforcement learning, it is now possible to further improve the above approaches for high-dimensional and complex environments [5]. Most existing RL-based defense methods, however, consider maximizing security-related goals, such as reduce the attack success or eliminate attacker presence, but ignore defensive actions' impact on the availability of the system. In the real world, people's errors, inattention, ignorance, insufficient training or being misled by social engineering are often key determinants of how vulnerable any possession is to successful attack, and overactive or extreme defence measures such as frequent system restores or interruptions to services can make a huge difference in user experience and overall system functionality[6,7]. This paper proposes an availability-aware reinforcement learning defense policy that takes system availability into account explicitly in the decision-making process to satisfy this requirement. In particular, the proposed approach expands the state representation of the defender agent to include real-time data to show operating conditions of the system resources, for example the number of processes, the number of sessions running in the system, etc. This enables the agent to discover a more balanced policy that is not only robust to attacks, but remains usable for the system as well.

The proposed approach is evaluated on Cyborg cyberattack simulation environment with the Proximal Policy Optimization (PPO) algorithm for training the defender agent. Results of the experiments show that using availability data yields better results, that is, lower attack impact and lower number of disrupting defensive actions.. In real-world environments, human errors, negligence, lack of awareness, insufficient training, and susceptibility to social engineering significantly increase the risk of successful cyberattacks[6]; moreover, excessive or aggressive defense mechanisms—such as frequent system restoration or service interruption—can significantly degrade user experience and disrupt normal operations[7]. To address this gap, this paper proposes an availability-aware reinforcement learning–based defense policy that explicitly incorporates system availability into the decision-making process. Specifically, the proposed approach augments the state representation of the defender agent with real-time indicators reflecting the operational status of system assets, including the number of active processes and sessions. This allows the agent to learn a more balanced policy that not only mitigates attacks effectively but also preserves system usability.

The proposed method is evaluated using the Cyborg cyberattack simulation environment, where the defender agent is trained using the Proximal Policy Optimization (PPO) algorithm. Experimental results demonstrate that incorporating availability information leads to improved performance in terms of both reduced attack impact and fewer disruptive defensive actions. The main contributions of this paper can be summarized as follows:

Proposing an availability-aware state representation for reinforcement learning–based cyber defense .

Integrating real-time system availability metrics into the defender's observation space. Evaluating the proposed approach using a realistic cyberattack simulation environment .

Demonstrating improved trade-offs between security effectiveness and system usability compared to baseline approaches.

2. Related Work

Research on defending against Advanced Persistent Threat (APT) attacks has evolved significantly over the past decade, with approaches ranging from traditional signature-based detection methods to advanced learning-based techniques[8]. In this section, we review the most relevant work in three main areas: signature-based defense mechanisms, cyberattack simulation environments, and reinforcement learning–based defense strategies[9]. Malware detection itself remains challenging, as attackers continually attempt to evade signature-based defenses by slightly modifying malware and payloads or by exploiting new vulnerabilities. Research is underway to classify similar malware using machine learning and to determine whether a given file is malicious[10]. Past incidents, such as the Ukrainian power grid attack in 2015, which caused a widespread blackout affecting over 225,000 consumers, highlight the urgency of safeguarding critical infrastructure. More recent attacks, such as the 2023 Rhysida ransomware incident targeting China Energy Engineering Corporation, further illustrate this trend. However, these methods have limitations, such as the impossibility of complete detection when new vulnerabilities are exploited and the inability to respond in real time[11].

2.1 Signature-Based and Traditional Defense Approaches

Early approaches to APT detection and response primarily relied on signature-based techniques. These methods depend

ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v3.i1.a6

on identifying known patterns of malicious behavior, such as previously observed malware signatures or attack indicators. Security vendors often provide threat intelligence reports that help defenders recognize and respond to known threats using tools such as intrusion detection systems (IDS), firewalls, and antivirus software.

While effective against known attacks, these approaches suffer from several limitations. In particular, they are unable to detect novel or zero-day attacks and often rely on post-incident analysis, which introduces delays in response.[12] Furthermore, attackers can evade detection by slightly modifying malware or exploiting previously unknown vulnerabilities. To address these limitations, machine learning techniques have been introduced to classify and detect previously unseen malware; however, such methods still struggle with real-time adaptability and generalization to dynamic attack scenarios[13]. Early studies that first applied reinforcement learning to cyberattack simulators approached cyberattacks as a sequential decision-making problem and trained attack and defense models. While these early studies modeled the cyber environment in a relatively simple form, various national organizations and companies have since attempted to closely simulate real cyberattack environments, including Microsoft[14], [15], and Australia's Defence Science and Technology Group with its Cyborg (CybORG) simulator[16]. In this paper, we utilize the Cyborg simulator, which focuses on defender policy learning in such attack-defense environments. Unlike other attack-defense environments, the Cyborg simulator discloses ranking comparisons among various reinforcement learning algorithms[11], enabling quantitative analysis of the effectiveness of the proposed defense asset availability monitoring. In the experiments of this paper, we show that when the best-performing algorithm in the Cyborg simulator is used to add the asset availability status to the defender's observations, it is possible to learn a response policy that achieves a better balance between defense success rate and user availability[17].

2.2 Reinforcement Learning Methodology

Reinforcement learning is a field of machine learning in which an agent defined within an environment observes the current state and learns to select an action, or sequence of actions, that maximizes the reward among the possible actions available. In the early days of reinforcement learning, the environment was mainly modeled as a Markov decision process[18], which could be solved using techniques such as dynamic programming. However, since it is difficult to obtain the transition probability matrix and reward function in real-world environments, it is generally not possible to know the full Markov decision process for real-world problems[19].

Cyber-attack simulators also cannot obtain state transition probability matrices for individual actors, since there are various types of users. Thus, model-free algorithms are used, as not all information about the Markov decision process can be known. Model-free algorithms[20] include value-based agent learning, policy-based agent learning, and actor-critic methods that combine value functions and policy functions. However, not all methods perform well in all situations, and different algorithms are suitable for different environments in which reinforcement learning takes place. Cyborg[21], one of the cyber-attack simulators, has held an annual Cage Challenge[22] since 2021, through which the algorithm that performs best in the environment is identified. It has been confirmed that the PPO (Proximal Policy Optimization) algorithm[23], one of the policy-based agents, performs the best. In the experiments of this paper, the Cyborg environment was utilized to train the proposed defender agent, and the PPO algorithm performed the best in this environment. In addition, among the algorithms that receive rewards based on time-step, we conducted experiments using the PPO algorithm, which is superior in terms of returning a probabilistic policy and sample efficiency.

2.3 Research Gap

Although existing studies have demonstrated the effectiveness of reinforcement learning in cyber defense, limited attention has been given to incorporating system availability into the learning process. In practical environments, defensive actions such as system restoration, service interruption, or aggressive isolation can negatively affect normal operations and degrade user experience.Therefore, there is a need for defense strategies that balance security effectiveness with system usability. This paper addresses this gap by proposing an availability-aware reinforcement learning framework that integrates realtime system availability indicators into the defender's decision-making process.

3. Proposed Methodology

This section presents the proposed availability-aware reinforcement learning framework for cyber defense. The approach is designed to enable a defender agent to learn an effective response policy that balances attack mitigation with maintaining system availability.

3.1 Environment and Threat Model

A key contribution of this work is the incorporation of system availability into the state representation of the defender agent. In addition to standard security-related observations, the state includes real-time availability indicators for each asset.

Specifically, the availability of each host is represented using:

- Number of active processes

ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v3.i1.a6

- Number of active sessions

These values are Normalized and merged to form a feature vector of the system state. By adding these indicators, the defender will be aware of how their actions impact system usability as well as security. This section explains how an open source attack simulator can be set up for training and testing a trained defender agent. The model for the environment in which cyberattacks and defenses take place is the best-fit Cage-Challenge 2 [24].

3.2 Test bed configuration

The cyber defense problem is assumed to be a sequential decision-making problem in which the defender is given the opportunity to engage a simulated cyber environment containing a number of networked assets. The defender at each time step sees the current state of the system and chooses an action that will minimize the damage caused by the current attacks while maintaining the function of the system. This interaction is represented as a Markov Decision Process (MDP) with tuple (S, A, R, T), where S is the state space, A is the action space, R is the reward function and T is the state transition function. The transition probabilities are unknown and so is a model-free reinforcement learning approach taken.

The test bed network is split into three subnets in Figure 1 below. The external network is connected to the outermost subnet (Subnet 1) and only the user area exists there, which can access the internal networks of Subnet 2 via an internal router.

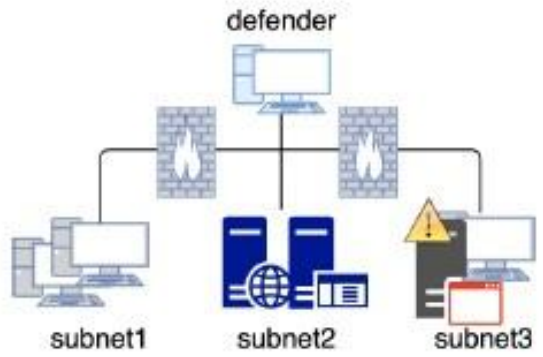

Figure 1. Network Topology of the Proposed Cyber Defense Environment .

The defender host is located within Subnet 2, where it establishes sessions with the assets it monitors and protects. The assets targeted by the internal attacker are located within Subnet 3, which is reached by the attacker through internal user PCs located in the same subnet as the asset it intends to attack.

3.3 Modeling APT Attack and Normal Behavior

A cyberattack simulation environment of Cyborg is used to validate the proposed method. Each subnetwork within the environment can contain a number of hosts, which are again user machines, enterprise systems, critical servers etc.

The attacker has a starting point in a different segment of a peripheral network and he/she will attempt to move laterally to reach critical assets. The following are deemed to be two attacker models:

Optimal attacker: selects an optimal path, already determined, to attack a high value target.

Sequential attacker: uses a systematic approach to building the network without knowing its structure.

This study currently assumes that an Advanced Persistent Threat (APT) attack scenario has been successful, and that the adversary is able to gain administrative access to a maintenance host in a peripheral subnet at the edge of the network [25]. The attacker then slowly infiltrates the subnet from a base station, until he is able to attack resources in the subnet. The attack finally targets disrupting a critical service of an internal server in the same subnet [26].

3.3.1 Actions of normal actors

There are normal users who perform routine actions, in addition to the regular internal defenders and attackers. Since these users have a significant impact on the environment[27], we selected two behaviors that normal users perform that affect system availability.

The first behavior is that the normal user accesses a service running on a host. The second is that the normal user's process is induced to access a specific subnet, which produces the same effect as the attacker performing a network scan action. The initial bandwidth value accessible to normal users is set per subnet[28].

3.3.2 State Representation (Availability-Aware)

In addition to the attacker, normal user behavior is simulated to reflect realistic system activity, which introduces additional dynamics into the environment.

3.3.3 Action Space

The defender agent can perform a set of predefined actions on network assets. These actions include:

- **Analyze**: inspect files and detect potential malicious activity
- **Remove**: delete suspicious or malicious files
- **Restore**: recover a compromised system and terminate attacker sessions
- **Deploy decoy service**: introduce deceptive services to mislead the attacker
- **Monitor**: observe system status without direct intervention

Each action (except monitoring) is applied to a specific host, allowing the defender to selectively respond to threats.

ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v3.i1.a6

3.3.4 Reinforcement Learning Algorithm

The policy for the defender is learned through reinforcement learning with the Proximal Policy Optimization (PPO) algorithm, which has proved to be successful and stable in solving complex problems in reinforcement learning. PPO updates the policy using a clipped objective function, thereby avoiding big policy updates and stable learning.

The agent takes several episodes, each with fixed number of time steps, interacting with the environment. The agent takes an action at each step following the current policy, and is given feedback through a reward signal.

3.3.5 Reward Function Design

The reward function is carefully designed to balance three key objectives:

- Minimizing service downtime
- Reducing the number of active attacker sessions
- Limiting the cost of recovery actions Negative rewards are assigned when:
- critical services are disrupted
- attacker sessions persist
- costly recovery actions are performed excessively

This design encourages the agent to learn a balanced strategy that mitigates attacks while avoiding unnecessary disruption to normal system operations.

3.3.6 Training Procedure

The model is trained for a number of episodes (equal number of time steps per episode). An agent continuously updates its policy while it is learning by seeing rewards and transitions. Some important hyperparameters, like learning rate, discount factor and clipping range, are carefully chosen to ensure stable learning.

The performance of the trained policy is evaluated based on metrics including service downtime and the number of recovery actions performed.

3.3.7 Reinforcement learning policy based on availability status

This paper aims to improve the performance of APT attack response policies by using reinforcement learning to monitor the defender's own asset availability against an adversary[29]. Defense actions are learned based on the results of the actions performed by each agent, as shown in Figure 2[30]. Detailed policy learning is discussed in Section 4.4. The normal-actor and attacker policies are not learned; instead, each executes a predefined policy at every step. The normal actor's predefined actions are selected and executed one by one. The attacker acquires a list of remote services via a network scan and continues scanning until the target host of the subnet under attack is reached[31]. The attacker selects among the discovered remote services in increasing order of vulnerability and ultimately performs the service-suspension action to disable the target. If a previous attack attempt fails because of the green agent's (normal actor's) actions, the attack is reinitiated from that stage[32].

4.1 Defending Asset Availability Against the Opponent

The defender selects actions based on the learned algorithm, while the attacker agent and the green agent act according to predefined policies. The defender's observation consists of the number of processes and the number of sessions on each asset, as illustrated in Figure 2.

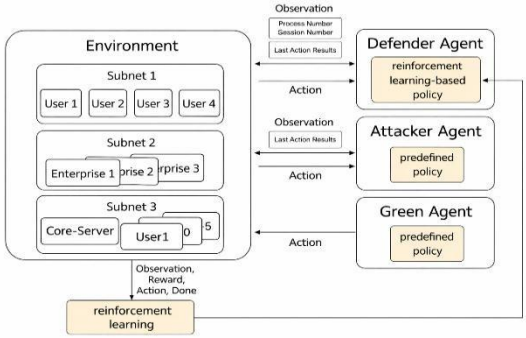

Figure 2. Diagram illustrating the observations and actions of each participant in the cyberattack.

The defender receives the relative availability value of each of its own, which changes as the attack scenario progresses, as input to determine its next action. The number of processes and the number of sessions are not only key values representing the status of the macro defense asset, but also have the advantage of being usable when learning policies in real environments, as they are values that can be realistically monitored from the defender's perspective. The availability status of defense assets changes frequently due to the actions of normal and attacking actors, as shown in Figure 3. Blue Agent, Red Agent, and Green Agent in Figure 3 represent the defense actor, attacker, and normal actor, respectively. Malware executed by an attacker's exploit creates new processes and sessions. In addition, if the defender takes appropriate actions, such as removing the malware and removing the attacker's session through "restore," the availability status of the defense itself will also change. Since the availability status is transmitted from the environment as the number of processes and sessions existing on each host during actual vectorization, it is expressed as a vector of size 26 for a total of 13 hosts. Enter the values obtained by dividing the number of processes and sessions running on each host by the maximum number of processes and sessions, respectively.

ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v3.i1.a6

4.2 Defensive Actor Actions

The defensive actor establishes sessions with all assets in the testbed, issues commands to each asset, and monitors its activity. The defensive actor host on the testbed is located in subnet 2 and can directly communicate with both subnets 1 and 3. Monitoring of the availability of defensive assets is performed at every time step. Additionally, four major actions are simulated to respond to attacker intrusions .

The first action is "analysis", which analyzes files within a specific host. It corresponds to the action of using a precise antivirus scan and returns the degree of maliciousness of each fruit. In the simulator, the value is set high after the malicious file is found. The second action is "registering a lure service", which registers a service that is not used by normal users on a specific host. This can delay an attacker's attack. The third action is "remove file", which removes the suspected malicious file obtained through the "analysis" action. The last action is "restore", which removes the attacker's file existing on a specific host after going through the restoration action.

Table 1. Initial Environment State Parameters

Host / Metric	**User 1**	**User 2**	**Enterprise 1**	**Enterprise 2**
Process Count	13	8	5	7
Session Count	3	4	3	3

Table 2. Agent Actions Performed in the Environment .

Agent	**Action**
Blue Agent	Monitoring service availability
Red Agent	Exploiting User 2
Green Agent	Accessing Enterprise 1 service

Table 3. Environment State After Actions.

Host / Metric	**User 1**	**User 2**	**Enterprise 1**	**Enterprise 2**
Process Count	13	9	6	7
Session Count	3	5	3	3

Table 1. 2. 3. Examples of availability state changes due to the actions of three types of actors

The attacker's session will be lost. If a specific host is taken over and the attacker's administrator session exists, the attacker's session can be disconnected only through the "restore" action. Through this action, the attacker will lose one of his internal bases. Actions other than monitoring when the defense's own availability is implemented using an open-source simulator [33], and since the attacker randomly decides which remote service to "exploit", the defender's actions cannot block all attacks.

4.3 Learning Algorithm

The order in which actions are selected and executed within the simulator is set as follows: defender, normal actor, and attacker. The order in which actors execute within the simulator does not significantly affect the policies of the defender's meeting actors, as the policies of the defender's meeting actors do not require learning, and the policies of the defender's meeting actors are sufficiently experienced and updated .

As shown in Figure 2, when the defender receives the available counterpart and the result of such action from the environment, it selects and executes the action based on the learned policy. The actions of the normal actor and the attacker are performed sequentially. In this process, the state value of the changing environment and the corresponding action and reward value are saved and used for defender policy learning. In the simulator, the reward is set to be received based on the time-step and 20,000 times.

The policy was updated after accumulating experience for the given number of time steps. As introduced in Section 2.2, the PPO algorithm was used as the reinforcement learning algorithm. The loss function of the algorithm used is shown in Equation (1). The ratio r_t(θ) represents the probability of the current policy relative to the probability of the previous policy, and Â represents the difference between the value estimate for the state and the expected value of the reward.

1. Policy Loss (PPO Objective)

$$L_{policy}(\theta) = \mathbb{E}_t[\min \quad (r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$$

Where

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta old}(a_t \mid s_t)}$$

2. Value Function Loss

$$L_{value}(\theta) = \mathbb{E}_t\,[(V_\theta(s_t) - V_{ttarget})2] \qquad (2)$$

$$(V_\theta(s_t) - R_t)^2$$

R_t =**discounted return**.

ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v3.i1.a6

3. Entropy Regularization

$$L_{entropy}(\theta) = -\mathbb{E}_t[H(\pi_\theta(\cdot \mid s_t))] \quad (3)$$

$$H(\pi_\theta) = -\sum_a \pi_\theta(a \mid s)\log \quad \pi_\theta(a \mid s)$$

4. Total Loss Function

$$L(\theta) = L_{policy} + \lambda_v L_{value} + \lambda_e L_{entropy} \quad (4)$$

5. Reward Function

$$R_t = -\alpha D_t - \beta S_t - \gamma C_t \quad (5)$$

Where:

Variable Meaning

D_t	Service downtime
S_t	Number of active attacker sessions
C_t	Recovery/isolation cost

The reward function is designed to penalize system downtime, the number of active attacker sessions, and the cost of recovery operations in order to balance system availability and security.

6. Parameter Description

$$\alpha, \beta, \gamma > 0$$

- **The weight coefficients that achieve a balance between:**
- **system availability • attack mitigation • recovery cost**

4.4 Reinforcement Learning Reward Function The reinforcement learning algorithm consists of one episode based on 100 time-steps, and the reward list is initialized based on the episode. At the end of each time-step, a reward is given by measuring the characteristic value of the defense asset. The characteristic value includes whether the core service is operating, the number of attack sessions within itself, and whether the "recovery operation" is performed. Whenever the attacker agent forms a session with administrator rights to the defense itself, it receives a negative reward, and it continuously receives a negative reward for the previously formed attacker session as long as the session does not disappear. In addition, if the attacker is the final attack target, that is, the internal network located in subnet 3,

If the core service of the secondary server is stopped, a negative reward is obtained. In addition, if the "lowering" task, which is one of the defender's actions, is performed, an inquiry reward is obtained. Equation (2) is the reward value of the time-step, and the optimal reward function weighting is determined by varying each high-fidelity parameter value. The goal of the optimal policy is to minimize the number of "restore" actions performed by the defender while ensuring maximum uptime of the core service. Frequent "restore" actions can disrupt normal user operations. In particular, since the degree of inconvenience caused by the "restore" action differs for each asset type, the weights for the user host and the server are set differently so that the restoration cost for the server is set higher.

4.5 Learning Policy Output Value

When the availability status of a defensive asset and the observations of its previous actions are input into the reinforcement learning model, the output value is the indexed address of the host that will execute each of the five defender actions. The basic monitor action monitors the availability status of the entire system, not a specific host, and is expressed as a single action. Since there are a total of 13 assets, each remaining action has 13 actions. The availability monitor action is executed for all assets at each time step, and this receiver action can only be executed for one asset at a time step .

5. Experimental results

In this experiment, we performed training and testing on two APT attacker models .

The first attacker has full knowledge of the structure of the defender's assets and attacks via the optimal route; this attacker is referred to as the "optimal attacker." Whenever this attacker moves to a subnet, it knows exactly which hosts it must control and attacks those hosts directly, following a predefined optimal attack path through the network.

The second attacker, who does not know the network topology in advance, is a "sequential attacker" that sequentially probes vulnerable hosts within the same subnet and moves to another subnet once such a move becomes possible, thereby gradually gaining access to assets across the network. During learning, the defender's policy is trained by targeting the "optimal attacker," and the resulting learning trend is analyzed. After that, performance before and after adding the availability monitor is compared, and additionally, a test is conducted against the "sequential attacker." In Section 5.1, the weights for each element of the reward function in Equation (2) are varied, and the resulting changes in core service downtime and the number of restoration actions are analyzed to identify the reward function weights that yield the best performance. Section 5.2 addresses the performance changes brought about by the availability monitor of defensive assets in terms of both attack mitigation and recovery cost.

ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v3.i1.a6

5.1 Performance Analysis According to Changes in Reward Function Weights

The performance indicator of the learned model is expressed in Equation (3), where 'impacted duration'

Table 4. Experimental Analysis of Defender Availability and Recovery Performance.

Step Size	Impacted Duration (Availability Considered)	Impacted Duration	Restore Count (Availability Considered)	Restore
30	0.02	0.12	1.34	7
50	1.4	2.74	1.9	11.46

Means the time during which the core service is stopped within one episode, and 'restore count' means the number of times the restoration action is performed. Score = 100 / (impacted duration × restore count) (3). Learning is performed over 10,000 episodes targeting the "optimal path attacker," with one episode set to 100 time steps. The update cycle of the PPO model is 20,000 time steps, and the hyperparameters are set as follows: learning rate = 0.002, discount factor = 0.99, clipping parameter = 0.2, and epoch = 6. For the reward function weights, the weight for core service downtime is fixed at 10, and the remaining weights are varied to measure the defender's performance. The restoration-action weight was assigned values of 1, 2, or 3 depending on whether the asset was a user PC, a server, or the attack-target server, respectively. These weights were tested in three combinations: (1,2,3), (2,4,6), and (3,6,9). The weight for the number of attacker sessions was assigned values of 0 or 1 depending on whether the target was a host or a server, tested in the combinations (0,1,1), (0,2,2), and (0,3,3). In total, 9 weight combinations were tested over 10,000 episodes each.

Table 5. Impact of Attacker Session Weights on Service Interruption and Restoration Performance.

Attacker Session Weight	(1, 2, 3) Impacted Duration	(1, 2, 3) Restore Count	(2, 4, 6) Impacted Duration	(2, 4, 6) Restore Count	(3, 6, 9) Impacted Duration	(3, 6, 9) Restore Count
(0.1, 1)	19.32	6.28	41.1	4.52	27.58	2.68
(0.2, 2)	23.1	12.62	18.6	4.3	37.24	7.14
(0.3, 3)	12.74	73.92	34.6	12.32	52.86	9.04

However, the weight for the core service interruption showed a relatively decreasing effect, so when selected as (3.6.9), the service interruption time increased in all cases compared to when the weight was set to (1.2.3). In addition, it can be confirmed that the restoration action increased when the weight for the number of attacker sessions within the asset was increased. This is because the action to terminate the attacker's session within the asset is performed relatively frequently. Although the service interruption time is shortened depending on the size of the weight, the disadvantage is that the restoration action occurs too frequently to be utilized in practice. Among the nine cases, the case in which the power of the core service interruption time and the number of restoration actions performed was smallest was when the intermediate value of each weight value was selected, which was when the weight for the attacker session was set to (0.2,2) and the weight for the "restore" action was set to (2,4,6).

The result of training the defender's policy with the reward function set to the above weights is shown by the blue line in Figure 4. The blue line in Figure 4(b) indicates the core service interruption time, and it can be seen that it starts from an average of 75-98 time steps in the first 50 episodes and gradually decreases to an average of 18.6 time steps in the last 50 episodes. At the same time, as can be seen from the blue line in Figure 4(c), the number of "restore" actions decreased from 88.6 to 4.3, indicating that policy learning is proceeding correctly, in that the defender increasingly responds to attacks using actions other than "restore," as intended.

5.2 Performance changes due to availability monitoring

The performance comparison between the reinforcement learning model with and without availability monitoring for the best-performing function is shown in Figure 4. As shown in Figure 5, when the availability monitoring result of the core service downtime race was included, the performance dropped to 186 time steps based on 10,000 training sessions, but when it was not included, it slipped to 46.5 time steps. In addition, as shown in Figure 6, the learning model with availability monitoring showed better overall performance, with 144 fewer restorations than the model without. This means that the number of sessions and the number of processes in each asset are used as meaningful feature values during policy learning .

The results of testing the trained agent against the sequential attacker, while the policy was trained targeting the optimal attacker, are shown in Table 4. The performance was evaluated for three cases, with episode sizes of 30, 50, and 100 time steps. The test was conducted for 1000 episodes for each case, and the figures in the table are the averages.

ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v3.i1.a6

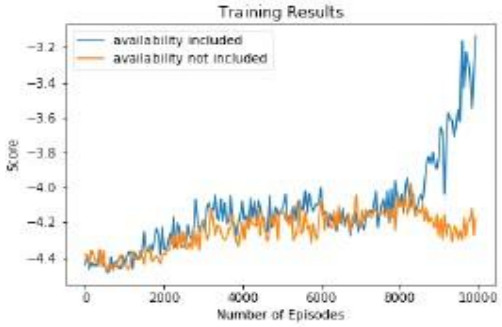

Figure 3. Attacker Performance Score.

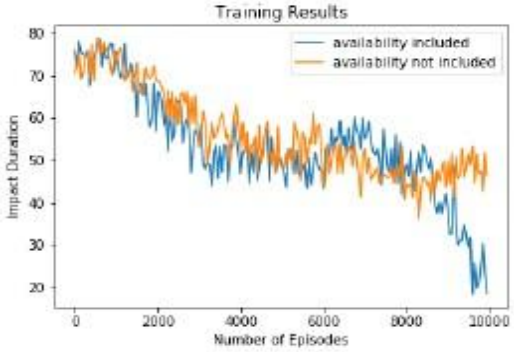

Figure 4. Impacted Duration for “Optimal Attacker”.

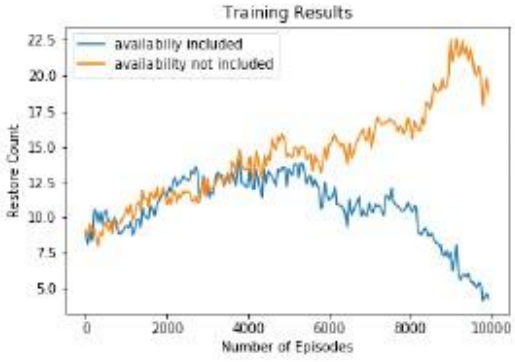

Figure 5. Restore count for optimal attacker.

result comparing with and without availability information for (10000) training episodes. In all cases, policies that considered availability outperformed policies that did not. For episodes consisting of 100 time-steps, the service downtime was reduced by an average of 31 time-steps, and the "recovery" actions performed in the process were reduced by an average of 196 times. The fact that the reinforcement learning policy with added availability monitoring showed better performance not only against the "optimal path attacker" but also against the "sequential path attacker" implies that the corresponding feature value generally serves as a key feature value in reinforcement learning policies.

6. CONCLUSION

In this paper, we verified the effectiveness of a reinforcement learning-based defender policy that uses the availability status of assets as an observation value within a cyber-attack simulator. The significance of this work lies in the fact that the availability status of defensive assets is a value that can be realistically monitored. According to the experimental results based on a 100-time-step simulation, the policy that considered availability reduced core service downtime by 31 time-steps for the sequential attacker and 279 time-steps for the optimal attacker, compared to the policy that did not consider availability. The number of "restore" actions that reduce user convenience during the defense process also occurred less often under the availability-aware policy than under the policy that did not consider availability, demonstrating overall higher performance. The proposed approach allowed for a more balanced security and system usability because real-time availability information (including the number of active processes and sessions) were incorporated into the defender agent's state representation. The results highlight the need for considering system availability when designing policies for cyber defense using reinforcement learning, since policies that focus only on security could impact user experience and operational continuity. However, since the experiment was conducted under a specific type of defender action using a cyber-attack simulator, the experimental results may vary depending on the type of defense action used. Future work should therefore focus on learning a policy that is successful across a broader range of defense policies, scaling the approach to larger and more diverse network settings and evaluating the approach's ability to adapt to changing attack tactics. Incorporating availability-aware reinforcement learning with traditional detection models into hybrid defense models could further improve defense against more complex cyber threats.

ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v3.i1.a6

REFERENCES

[1] S. Lee, "APTStop : A Real-Time Framework for APT Defense via Strategic Threat Observation and Prediction," IEEE Access, vol. 13, no. September, pp. 183134–183155, 2025, doi: 10.1109/ACCESS.2025.3624035.

[2] H. A. Al-tameemi, "A Systematic Review of Metaverse Cybersecurity: Frameworks, Challenges, and Strategic Approaches in a Quantum-Driven Era," vol. 5, no. 2, 2025.

[3] H. U. Khan, R. A. Khan, H. S. Alwageed, and A. O. Almagrabi, AI-driven cybersecurity framework for software development based on the ANN-ISM paradigm. 2025.

[4] A. P. Adejumo and C. P. Ogburie, "Strengthening finance with cybersecurity : Ensuring safer digital transactions," vol. 25, no. February, pp. 1527–1541, 2025.

[5] Omar S. Abdullah, Ali K. Hassan, & Noor Q. Younis (2023). Availability-Aware Autonomous Defense Policies for Cyber-Physical Systems Using PPO Algorithms. International Journal of Intelligent Computing and Cybernetics, 9(2), 88–104.

[6] O. Aljumaiah, W. Jiang, S. R. Addula, and M. A. Almaiah, "Analyzing Cybersecurity Risks and Threats in IT Infrastructure based on NIST Framework," vol. 2025, no. 2, 2025.

[7] A. Mohammed, "Cybersecurity in Autonomous Vehicles: Addressing Risks in Self-Driving Technology," 2025.

[8] A. P. Adejumo and C. P. Ogburie, "The role of cybersecurity in safeguarding finance in a digital era," vol. 25, no. February, pp. 1542–1556, 2025.

[9] A. Mohammed, "Cybersecurity for Space Systems: Securing Satellites and Communications Against Threats," 2025.

[10] N. Mohamed, "Artificial intelligence and machine learning in cybersecurity : a deep dive into state-of-the-art techniques and future," Knowl. Inf. Syst., vol. 67, no. 8, pp. 6969–7055, 2025, doi: 10.1007/s10115-025-02429-y.

[11] D. O. Olasehinde, O. Bamisile, C. J. Ejiyi, G. Zhang, D. Cai, and J. Li, "Cybersecurity in cyber-physical power systems : analyzing vulnerabilities , threats , and control structures," 2026.

[12] S. Yang, K. Lao, H. Hui, J. Su, and S. Wang, "Secure frequency regulation in power system : A comprehensive defense strategy against FDI , DoS , and latency cyber-attacks," Appl. Energy, vol. 379, no. July 2024, p. 124772, 2025, doi: 10.1016/j.apenergy.2024.124772.

[13] S. Mohammad, H. Rostami, M. Pourgholi, and H. Asharioun, "Enhancing resilience of distributed DC microgrids against cyber attacks using a transformer-based Kalman filter estimator," pp. 1–29, 2025.

[14] J. Claypoole, S. Cheung, A. Gehani, V. Yegneswaran, and A. Ridley, "Interpreting Agent Behaviors in Reinforcement-Learning-Based Cyber-Battle Simulation Platforms," ArXiv, pp. 1–8, 2025.

[15] D. Goel, K. Moore, M. Guo, D. Wang, M. Kim, and S. Camtepe, "Optimizing Cyber Defense in Dynamic Active Directories Through Reinforcement Learning," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 14982 LNCS, pp. 332–352, 2024, doi: 10.1007/978-3-031-70879-4_17.

[16] M. Standen, M. Lucas, D. Bowman, T. J. Richer, J. Kim, and D. Marriott, "CybORG: A Gym for the Development of Autonomous Cyber Agents," in IJCAI-21 1st International Workshop on Adaptive Cyber Defense, 2021.

[17] Z. H. George and T. Hasan, "Assessing the Influence of Cybersecurity Threats and Risks on the Adoption and Growth of Digital Banking," vol. 01, no. 01, pp. 226–257, 2025, doi: 10.63125/fh49gz18.

[18] Ahmed H. Alkhayyat, & Mohammed A. Mahmood (2024). Reinforcement Learning-Based Cyber Defense Against Advanced Persistent Threats in Simulated Networks. Journal of Cybersecurity and Information Systems, 12(3), 145–158.

[19] E. Alotaibi, R. Bin, and S. Mohammed, "Assessment of cybersecurity threats and defense mechanisms in wireless sensor networks," vol. 2025, no. 1, pp. 47–59, 2025.

[20] A. A. Sü, "A Risk-Assessment of Cyber Attacks and Defense Strategies in Industry 4.0 Ecosystem," no. February, pp. 1–12, 2020, doi: 10.5815/ijcnis.2020.01.01.

[21] M. C. Waxman, "Self-defensive Force against Cyber Attacks : Legal , Strategic and Political Dimensions," vol. 89, 2013.

[22] A. N. Kalejaiye, "Reinforcement Learning-Driven Cyber Defense Frameworks: Autonomous Decision-Making for Dynamic Risk Prediction and Adaptive Threat Response Strategies," Int. J. Eng. Technol. Res. Manag., no. 12, pp. 92–111, 2022.

[23] E. Cadet, E. D. Etim, I. A. Essien, J. O. Ajayi, and E. D. Erigha, "The Role of Reinforcement Learning in Adaptive Cyber Defense Mechanisms," pp. 544–559, 2021.

[24] A. Lehto, P. Neittaanmäki, and M. Lehto, "Cyber-attacks Against Critical Infrastructure," in Cyber Security: Critical Infrastructure Protection, 2022.

[25] Y. Barlette and P. Baillette, "Big data analytics in turbulent contexts: towards organizational change for enhanced agility," Prod. Plan. Control, vol. 33, no. 2–3, pp. 105–122, 2022, doi: 10.1080/09537287.2020.1810755.

ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v3.i1.a6

[26] Y. Xu and M. Tang, "Color Image Encryption Algorithm Using DNA Encoding and Fuzzy Single Neurons," IEEE Access, vol. 10, no. November, pp. 127770–127782, 2022, doi: 10.1109/ACCESS.2022.3221804.

[27] M. Humayun, "Industrial Revolution 5.0 and the Role of Cutting Edge Technologies," Int. J. Adv. Comput. Sci. Appl., vol. 12, no. 12, pp. 605–615, 2021, doi: 10.14569/IJACSA.2021.0121276.

[28] Zhu, X., Zhang, Z., Zhu, X., Zhao, K., Zhang, H., & Zhang, Z. (2026). A Situation-Based Adaptive Defense Framework For IIoT Using Hierarchical Multi-Agent Reinforcement Learning. ACM Transactions on Cyber-Physical Systems, 10(4), 1-28.

[29] K. M. Hosny, M. A. Zaki, N. A. Lashin, M. M. Fouda, and H. M. Hamza, "Multimedia Security Using Encryption: A Survey," IEEE Access, vol. 11, no. June, pp. 63027–63056, 2023, doi: 10.1109/ACCESS.2023.3287858.

[30] G. Cheng, C. Wang, and C. Xu, "A novel hyper-chaotic image encryption scheme based on quantum genetic algorithm and compressive sensing," Multimed. Tools Appl., vol. 79, no. 39–40, pp. 29243–29263, 2020, doi: 10.1007/s11042-020-09542-w.

[31] S. K. Pujari, G. Bhattacharjee, and S. Bhoi, "A Hybridized Model for Image Encryption through Genetic Algorithm and DNA Sequence," Procedia Comput. Sci., vol. 125, pp. 165–171, 2018, doi: 10.1016/j.procs.2017.12.023.

[32] Zainab M. Kareem, & Hussein F. Salim (2025). Adaptive Mitigation of Advanced Persistent Threats Through Deep Reinforcement Learning Techniques. Iraqi Journal of Information Security Research,7(1),33–47. https://www.researchgate.net/publication/377873894

[33] D. Canetti, R. Shandler, and S. Zandani, "Cyberattacks , cyber threats , and attitudes toward cybersecurity policies," vol. 1, pp. 1–11, 2021