applied sciences

Article

Cyber Environment Test Framework for Simulating Command and Control Attack Methods with Reinforcement Learning

Minki Jeong [1], Jongyoul Park [1] and Sang Ho Oh [2,*]

[1] Department of Applied Artificial Intelligence, Seoul National University of Science and Technology, Seoul 01811, Republic of Korea; hoodit@seoultech.ac.kr (M.J.); jongyoul@seoultech.ac.kr (J.P.)

[2] Department of Computer Engineering and Artificial Intelligence, Pukyong National University, Busan 48513, Republic of Korea

* Correspondence: shoh0320@pknu.ac.kr; Tel.: +82-51-629-6108

Abstract: Recently, the IT industry has become larger, and cloud service has rapidly increased; thus cybersecurity to protect sensitive data from attacks has become an important factor. However, cloud services have become larger, making the surface area larger, and a complex cyber environment leads to difficulty managing and defending. With the rise of artificial intelligence, applying artificial intelligence to a cyber environment to automatically detect and respond to cyberattacks has begun to get attention. In order to apply artificial intelligence in cyber environments, a simulation framework that is easily applicable and can represent real situations well is needed. In this study, we introduce the framework Cyber Environment (CYE) that provides useful components that abstract complex and large cloud environments. Additionally, we use CYE to reproduce real-world situations into the scenario and apply reinforcement learning for training automated intelligence defense agents.

Keywords: cyber security; reinforcement learning; cloud service

Academic Editors: George Drosatos and Konstantinos Rantos

Received: 14 January 2025
Revised: 13 February 2025
Accepted: 14 February 2025
Published: 17 February 2025

Citation: Jeong, M.; Park, J.; Oh, S.H. Cyber Environment Test Framework for Simulating Command and Control Attack Methods with Reinforcement Learning. *Appl. Sci.* **2025**, *15*, 2120. https://doi.org/10.3390/app15042120

1. Introduction

Data may contain important information such as personal or secret information e.g., weight, bank accounts, corporate quality, or blueprint [1]. Storage and sharing of these data required a lot of storage space and a lot of transmission time. In the process of storing and sharing such data, the emergence of networking was inevitable [2]. Networking allowed many nodes to communicate and work with data and store it beyond its physical limits. Advances in the IT industry have made it possible to share and store data beyond previous physical and temporal limits. Based on this convenience and accessibility, the cloud IT industry has grown dramatically. More and more services have emerged in digital for this accessibility, and many companies and organizations have begun to digitally store and share information. Due to data sharing via a network, so new type of threat is appeared called cyber threat [3]. Cyber threat is any type of situation or event that negatively affects these services, data, etc. such as unauthorized access, data modification, leakage and removal, physical destruction, and out-of-service [4]. Among these threats, intentional cyberattacks have negative effects, such as financial, leaking valuable data from targets, or destroying services for purposes such as obstructing competitors. With the advent of cyber security to protect data from these cyberattacks, it was inevitable for defense and attack teams to compete [5,6]. As the IT industry grows, keeping more data and more complex services safe from attacks has fatigued defense teams, and with time, attackers begin to attack using more advanced and sophisticated technologies [7]. Especially, attacks in the past have begun using more sophisticated methods, unlike indiscriminate multiple attacks on a wide range [8,9]. An attack called Advanced Persistent Threat (APT) prepares a specialized and sophisticated attack by specifying a target. Additionally, APT also uses a variety of methods, which use social engineering techniques as well as software engineering to achieve set goals [10]. In order to defend against these APTs, a strategy called Cyber kill chain, which is derived from kill chain, emerged. The kill chain is a military term that has a phase of identifying targets, establishing operations, and attacking targets. The cyber kill chain is also a model that divides cyber threats into several stages and focuses on each stage to detect and respond to them. Cyber kill chain divides attack into 7 steps, analyzes each step, and aims to prevent attacks as early as possible. These 7 divided steps are as follows [11]. The first step is reconnaissance. In the reconnaissance step, collect the target's system information. These data include email, used OS type or version of the program, etc. Next step, Weaponization: Find vulnerabilities based on previously collected information. With these vulnerabilities, weaponize it and bundle it in payload. Delivery is the third step: With the weaponized bundle, the attacker delivers the bundle to the victim via a phishing email, USB, etc. The fourth step is Exploitation: In the exploitation step, the attacker's weaponized bundle exploits a vulnerability to execute code on the victim's systems. After exploitation, Installation: Weaponized bundle installed on victim's system. The next step is Command and Control: The attacker remotes the victim's system via command and control. Finally, Actions on objectives are the last step. In this step, an attacker can achieve their original goal. Exfiltrate data, credential access or hopping to another network, etc. Despite the efforts of these defenders, the growing and complex cloud environment has become difficult to defend against for reasons such as its wider exposure network and complex configuration and has caused mistakes. Additionally, the magnitude of the attack has evolved to a massive scale unlike in the past [12]. In this situation, it was natural for defenders to build automated defense and detection systems. One of the types of detection systems, the signature-based network intrusion detection system, detects malicious traffic recorded in the database but is difficult to detect since it does not record patterns that change with rapidly evolving attacks. In addition, the complexity and quantity of existing methods have been limited by the increasing IT industry. To break through the limitations of these existing methods, the detection system has evolved once with the application of machine learning. There is a study [4] that analyzes intrusion detection by analyzing window register access with machine learning techniques. In another study [13], machine learning algorithms were used to detect DDOS attacks by analyzing various features to detect various DDOS attacks by increasing scalability. Advances in artificial intelligence and interest in it have presented a new direction for applying deep learning algorithms using deep neural networks. This study [14], proposed a method for detecting cyberattacks against autonomous vehicles using deep learning algorithms. Algorithms developed by analyzing features of certain datasets achieved fast and high performance with limited resources. In addition, the application of reinforcement learning was also studied because several indicators in these intrusion detection tasks can be considered as scores. This study [15], achieved high accuracy by developing algorithms to detect abnormal flows using reinforcement learning. RL, which has a structure that interacts with the environment, earns rewards, and learns, has achieved great success in chess games and has increased interest in autonomous agents using artificial intelligence, which has shown the potential of agents interacting in network environments, not just flow detection. Training automation agents in cyber environments has emerged in cybersecurity. Various frameworks have been proposed to create virtual cyber environments and train agents operating on them, but there have been limitations, such as limitations on scenario configuration. So, we introduce CYE. CYE can write scenarios based on code, allowing for more flexible configuration and more complex simulation environments with nodes that communicate with various processes. CYE also delays communication between nodes, enabling the simulation of delayed situations. The following section will introduce the frameworks that can make up network environments and agents.

2. Related Works

There are several frameworks for building and simulating a cyber environment. CyberBattleSim [16] uses the high-level abstraction of computer networks so that researchers can easily build networks and learn automated agents. By providing various goals and configurable nodes, the automated agent can learn from each node's state and goal of the scenario. High-abstracted observation provides information like credentials and OS information, and the action being executed is determined by the status of the host and the appropriateness of the action. FARLAND [17] provides a sophisticated network-based simulation environment. The information gathered from the FARLAND simulation is high fidelity and also provides an environment that is friendly for training AI agents. However, FARLAND does not support simulating the host state. CYbORG [18] can build a simulation with YAML file configuration and simulate. In CYbORG each host has its own state machine that includes network state, host OS information, process, etc. In this environment, the action has a result value that is executed based on the state of the hosts. However, the state machines provided by CYbORG make it difficult to track the interactions between processes. Furthermore, while CYbORG supports configuration network topology, the delay that occurs when data are transferred from the network is difficult to model. Nasimemu [19] provides high-fidelity simulation and emulation environments based on the Network Attack Simulator. The simulation environment supports modeling conditions and probability of action being executed and affects to environment based on abstract states like host, service, subnet, network topology, and action such as network scan, privilege escalation, etc. Additionally, Nasimemu provides a test environment by emulating this environment with virtual machines. PenGym [20] provides higher fidelity with simulations of Nasimemu using virtualized machines and real-world attack tools wrapped in Python. These environments have two features. The first feature is Scenario file-based. Scenario file-based has the advantage of being able to write scenarios easily and quickly, but it has the disadvantage of being able to model only the actions specified in the process of simulating a scenario. This method is easy to model popular and widely used programs, but it is difficult to simulate programs or methods that are not supported in scenarios. The second feature is Higher fidelity makes implementation more complex. High-fidelity simulations or emulations are not easy to integrate with RL agents. Environments like PenGym use libraries wrapped in Python to run programs like Metasploit. However, researchers might want to model custom programs that do not support in framework. At this point, you need to write additional code, wrap it in Python, and add it to the framework. However, traditional libraries are not easy to configure. For example, YAWNING Titan rather provides a higher level of abstraction. At the Yawning Titan, researchers focused on network structures rather than detailed host states, focusing on these structures and developing and testing algorithms. So, we introduce CYE. CYE provides tools that make it easy to build a finite state machine and physical network device to build a flexible network topology. Additionally, code-based scenario configuration allows users to freely write state and policy with Python code to overcome the restrictions from existing scenario-based environment configuration. This feature allows researchers to adjust the degree of abstraction of simulations and implement flexible attack and defense methods. CYE is made of pure Python only, making it easy to debug, and it is lightweight and easy to combine with other programs since it does not require a virtual machine.

3. Cyber Environment: CYE

With the above limitations, we propose a new cyber environment simulation framework, CYber Environment (CYE). CYE is a highly abstracted cyber environment simulation tool that can configure cyber threats and defense situations. In CYE, each node has an operating system (OS) that manages computer resources and processes so that you can feel free to create complex inner processes, communication, and node interactions. To support building complex interactions, CYE provides useful components and functions to build complex networks and processes.

3.1. CYE Design

Let $\mathcal{G}_{net} = \langle V_{net},\ E_{net} \rangle$, $\mathcal{G}_{net}$ is an entire network in the environment. V_{net} is a set of nodes, with E_{net} represents relation between each node. ith node in set $v_i \in V_{net}$ has its own state S^{v_i} corresponding to time t, $S_t^{v_i}$. Each node has an internal policy that the node has $\mathcal{P}^{v_i}$—can be deterministic and probabilistic. Node's state is updated to $t+1$ node state by the update function $U^{\mathcal{P}^{v_i}}$ derived from the policy $\mathcal{P}^{v_i}$, so that $S_{t+1}^{v_i} = U^{\mathcal{P}^{v_i}}\left(S_t^{v_i}\right)$. However, this expression has the condition that the node v_i updated only by its internal state independently without any relation to other nodes $v_j \in V_{net}(j \neq i)$. However, nodes interact with other nodes and state updated by not only internal state but also external nodes. Node can receive signal from other node via network communication and also can send signal to another node e.g., ping to specific IP address. This means when node state in time t, $S_t^{v_i}$ update is determined by internal state, policy and also affected by other nodes state, policy $S_t^{v_j},\ \mathcal{P}^{v_i},\ v_j \in V_{net}$. Therefore $S_{t+1}^{v_i}$ is determined by entire graph state $S_t^{\mathcal{G}}$ and policy $\mathcal{P}^{\mathcal{G}}$ that aggregates each node's state and policy. So, node state update by $S_{t+1}^{\mathcal{G}} = U^{\mathcal{P}^{\mathcal{G}}}\left(S_t^{\mathcal{G}}\right)$. However, figure out entire network $S_t^{\mathcal{G}}$, $\mathcal{P}^{\mathcal{G}}$ is very difficult or impossible since network is too large or inaccessible, such as a private network for only allowed user. According to these limitations, our actual observation state S_t^{O} is limited region of $S_t^{\mathcal{G}}$, so we can consider $\mathcal{F}_O$ as feature extractor from $S_t^{\mathcal{G}}$, so that $S_t^{O} = \mathcal{F}_O\left(S_t^{\mathcal{G}}\right)$. Ideally, since the next state of a node is determined by its previous states and action vector, it can be considered as a Markov model especially Markov decision process. However, as we mentioned before, the space we can observe is limited so that we can consider as Partially Observable Markov Decision Process (POMDP). One thing what we have to remind is where we are interested in some specific states S_t^{I} derived—like network vulnerability score, specific node's state—rather than the entire network state $S_t^{\mathcal{G}}$. So, CYE provides useful kits to modeling $\mathcal{G}_{net} = \langle V_{net},\ E_{net} \rangle$ that represent abstracted network topology, nodes that have $S_t^{v_i}$ so that makes it possible to model complex state machine. Since flexible environment can be modeled, researchers feel free to restrict or expand observation state S_t^{O} by configuring $\mathcal{F}_O$. Reward and goal derived from interested state S_t^{I} are also free to be set. The below section briefly describes CYE design for helping modeling networks.

In CYE, there are two types of nodes: network type node and compute type node, and the node type indicates which task the node is focused on. Network type node focuses on connecting the networks and helps configure the entire network topology such as switch and router. Compute type node focuses on representing complex internal state machines and communicating with internal and external state machines such as PC. Note that, a network-type node also has its own state like a routing rule that identifies which port to send, so the node focuses on networking not only. CYE abstracts Open Systems Interconnection Reference Model (OSI) 7 layers, since CYE is focused on exchanging information between nodes not representing real-world data transmitted. The following section will introduce three nodes that help to model network.

3.2. Node Types

3.2.1. Computer Type Node—PC

PC is a node that has an operating system that provides a complex internal state. OS helps manage and communicate between software and hardware easily, such as file systems, network communication, and process management. In CYE, it abstracts OS's process and network communication. The process is an object that can be executed on the specific OS in CYE. This process object can have its own state and policy. In more detail, the OS receives step calls from the environment. The OS propagates a step call to the NIC if exists so that the OS can receive packets from the NIC. After the NIC step propagation is completed, the OS starts propagating the step sequentially to the processes. Processes can have executable policies that run at every step by implementing the step method. The process can also perform OS-related tasks, such as network operation or process creation via useful functions provided by OS, such as system call e.g., self.os.create_new_proc(). In CYE, network-related tasks use a unit called a packet, which is much different from the meaning of a real-world packet. The packet contains the source and destination for the IP/MAC address to identify source and destination nodes. The rest of the information is contained in a field that can store dynamic data. In other words, except for the information that can identify the source and destination node, the rest of the information is contained in the payload, unlike the actual network communication where the information is encoded and decoded in bit. Every network communication in CYE is transmitted between nodes only once every step.

3.2.2. Network Type Node—L2Switch

L2 switch is used when connects more than two nodes. L2switch sends packets to the right outbound port via the MAC address in packet. The internal state of the switch is the MAC address table. When switching, L2switch broadcasts the packet if there is no MAC address in the MAC address table. It also writes the MAC address and port in the MAC address table when a packet is received from a specific port, and when sending a packet, query the MAC address table and if there is a MAC address and outbound port is written then send to that port.

3.2.3. Network Type Node—Router

Router used when connecting network. The router determines where to end the packet with the packet's IP address. Usually, a router is used to connect the internal network to the external network, where the router helps to specify and configure the scope of the subnetwork. The router has an ARP table and a routing rule for the internal state. The ARP table matches the IP and MAC address table. This allows the router to obtain the IP address of the packet and know to which port the packet should be sent. The routing rule is about which network to send the packet by looking at the destination address. These rules are written in the routing table and the router sends packets to where it should be sent from rules.

Using these provided nodes, researchers can model complex networks, create various processes, and articulate their complex relationships in an abstract way. Additionally, these structures allow us to freely control the range of states we want to observe and also freely organize the objectives and rewards of the environment with pure Python methods.

4. Scenario Configuration

We configured the environment with CYE and trained the agent with the scenario. To create a scenario that has complex interaction between nodes, we searched the actual cyber threat case by Admin@338 [21]. Admin@338 tries to get initial access via phishing email and installing malware. After getting initial access and installing malware, the malware communicates with the red team server and exfiltrates the host's information. One thing to look at carefully in this attack technique is that malware used legitimate cloud services such as CnC servers to communicate with red team attackers and take additional processes and processes in the host communicated with internal processes and OS. In this scenario, we are inspired by these attack techniques, and design abstracted scenarios by dividing them into red team, blue team, and green team. The below section will explain attack techniques with MITRE ATT&CK matrix [22] and scenario configuration.

4.1. Tactics

4.1.1. Phishing Email [T1566.001]

A phishing email is a tactic to get initial access to the target host. admin@338 using the email that attaches a malicious file that lures victims to launch it. If the victim host launches a malicious file, then the malware will try to launch. There is some probability that malware does not work due to the runtime environment. To represent phishing emails and launch malicious files and successfully run on the victim's host, simulation abstracts these steps to malware runs on the victim's host in a probabilistic manner.

4.1.2. Command and Control [TA0011]

Command and Control (CnC) is a tactic to communicate with compromised victim hosts and control them. After compromising the victim host, adversaries try to communicate with the victim to do additional actions. Admin@338 using Dropbox is a legitimate cloud service to communicate with compromised hosts. Since Dropbox is a legitimate cloud service that has many users, adversaries use Dropbox to communicate secretly with compromised hosts to try to avoid surveillance security solutions. To represent this tactic, in the scenario, the malware communicates with the red team via a legitimate cloud service so that the blue team agent network packet scanning can not detect malware communication.

4.2. Team

Building simulation based on these tactics, we make three groups that work on the network, Green, Blue, and Red. The goal of this scenario is to get as high a score as possible. The below section will explain each node's role in this scenario.

4.2.1. Green Team

The green team is similar to employees in the company. Creating valuable files and send to the blue team. After the blue team receives these files, getting scores as much as it is worth. In this simulation green team creates random valuable files continuously 20 steps after the process started. Valuable file has reward scores between 30 and 50. These files need time to complete depending on their value, and use file write system calls every step until complete.

4.2.2. Blue Team

The blue team is similar to the centralized security team in the company. Collecting green host states and determining which action is right. Blue team has three processes, Collecting valuable file process, blue team security process, and blue team security server process. Collecting valuable files process open server and listen to receive valuable file created by green team. When the valuable file reaches to blue team server, then update the global reward as much as the valuable score in the file. Each green team host has a blue team security process for collecting, sending, and running operations. Reflecting the inability to gather all information, the security process will collect limited information about hosts that might be helpful in diagnosing the host's state. Additionally, get operations from the blue team server and execute the right actions. To show flexible and abstracted OS configuration, the blue team process collects system call events. The following lists show the data that the security process collects:

- Count of file event (hook kernel event related to file)
- Count of process events (hook kernel event related to the kernel)
- Count of network events (hook kernel event related to socket)
- Network scanning result. True if scanning result exists

Security process actions. All actions are executed following operations.

- Sleep: doing nothing
- Process scanning: Scanning processes. If the malicious process is detected then kill that process.
- Network scan: Scanning network packets. The scanning result is True if the packet can be scanned (not encrypted) and has malicious contents.
- Reset host: Reset the host's PC. OS became initialized. All processes and files are removed and the new system will be installed on the PC.

The blue team security server receives the host's state and decides which action is appropriate for the host. Agent observes restricted and delayed host's state and learns an appropriate policy to get a higher score. Since the security server collects data from green team hosts and sends operation, there is observation and action delay. So, in this scenario, we average observation space until blue team action occurs and receive a reply from the target host.

4.2.3. Red Team

The red team is an adversary and tries to compromise and exfiltrate valuable files. The red team gets rewards if successfully exfiltrates valuable files or the host's information. To take over valuable files, red team uses red team malware, red team backdoor, red team CnC server, red team backdoor server. The red team uses phishing emails to get initial access to the green team host. The red team uses legitimate cloud service as CnC server to communicate and get operations. Since CnC server uses a legitimate cloud service, so hard to detect malware communication. After the malware successfully communicates with the red team, probabilistically, execute the next operation by installing a backdoor. If the backdoor launches successfully, start to hooking file create an event, and send it to the red team backdoor server. If a valuable file is hijacked and successfully sent to the backdoor server and received, then the red team will update the global reward with the -4 * value of the file. Red team malware communicates with the red team via CnC server. After the malware is installed, the malware communicates continuously to CnC server and gets the operation code which action will be executed. Malware has four operations. 1. Send live code to notify to red team CnC server. 2. List up files and send the filename list to the server. 3. Exfiltrate target file. 4. Download the backdoor and execute it. Assume that the malware CnC server is a legitimate cloud service so that it is not easy to detect via network scanning. So, malware only can be detected by process scanning. Red team backdoor is a backdoor, that continuously takes over files and sends them to the red team backdoor server. Backdoor uses the red team backdoor server which is not a legitimate cloud server or encrypted protocol to send files, so blue team can detect backdoor communication with network scanning. Usually, the backdoor is not easy to remove, so the blue team process scanning cannot detect and remove the backdoor. Therefore, the blue team agent must use reset action to remove and recover the green team host. Figures 1 and 2 show observation space and reward in this scenario.

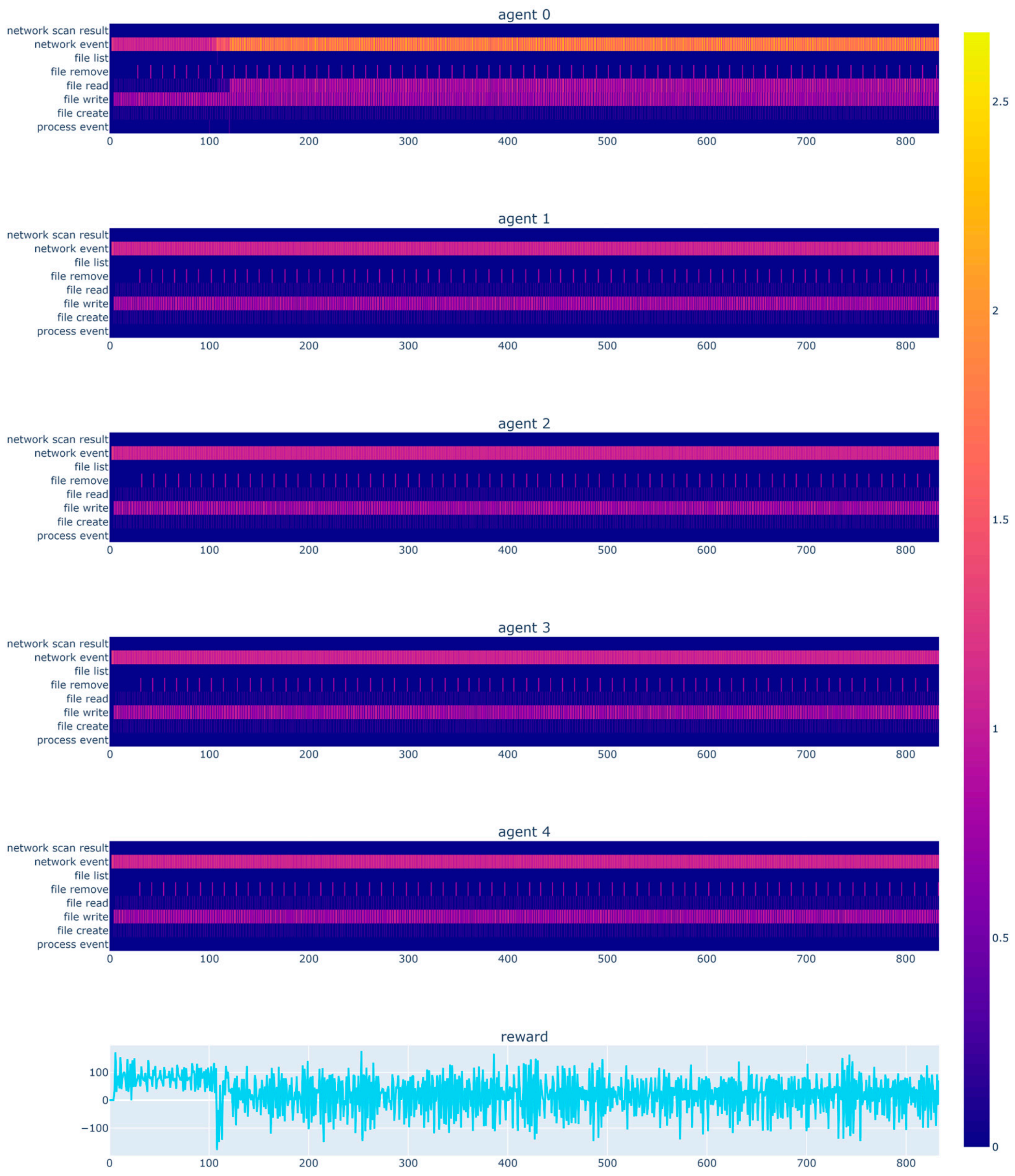


Figure 1. Visualization figure of blue team observation space and reward.

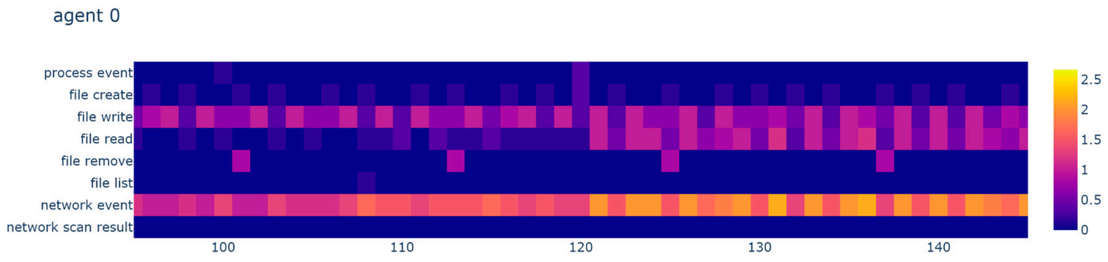


Figure 2. This figure shows visualization figure of agent 0's observation space when compromising agent 0 at 100th step. Malware send file list to red team server and red team CnC server receives list and exfiltrates files. After exfiltrate few files, CnC server deploy backdoor to compromised agent and backdoor continuously exfiltrates files and sends them to red team backdoor server.

5. Deep Q-Learning (DQL)

To train our blue team agent, we use the RL approach. Simulation space has limited observation space, action, and reward so with the RL method blue team agent can learn proper action via observation. Our environment can be represented with an action-state pair, when blue team agents act, green and red team agents will be affected. So, we use Q-learning to learn proper action. Q-learning is the RL method for learning how valuable state-action pairs (s_t, a_t) called Q-value. To estimate the Q-value properly, update the Q-value from the next steps and reward. To reflect further values, use gamma to cumulate and regulate further values. This factor is very important since in our environment blue team agent can only observe a delayed state from the green team agent.

$$Q(s_t, a_t) \leftarrow r_t + \gamma \bullet \max_a Q(s_{t+1}, a) \tag{1}$$

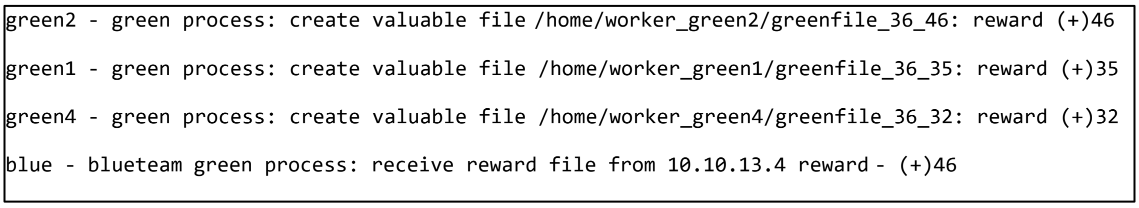
In this environment, the observation state has an infinite range and is complex. So, to estimate the Q-value, we use a deep Q-network that uses a neural network. Q-network will work well on continuous space also smoothing unobserved Q-value and very effective with complex states. However, exploiting with max Q-value can cause local minimum so that can not find optimal Q-value. To learn various states, the agent needs to explore and update Q-value. To explore various spaces that are against greedy action, the agent explores with probability ϵ. Starting with a large value of ϵ, explore actions, and gradually lower ϵ to adjust the agent to exploit and learn the appropriate behavior. In our setting, the deep neural network has four hidden layers. We use AdamW optimizer with $\beta_1 = 0.9$, $\beta_1 = 0.999$ and learning rate 1e−4. Additionally, we use Huber loss to be robust when the Q estimation loss is large. The hyperparameter to train the DQL agent follows Table 1.

Table 1. Hyper Parameter to Train DQL Agent.

Hyper Parameter	Value	Description
Discount factor	0.99	Discount factor (gamma) to discount cumulate reward
Stepper episode	40,000	Number of Scenarios
Stack frame	4	Stack recent frames and input to Q network
Update rate	0.005	Soft update network factor. $\theta' \leftarrow \tau\theta + (1 - \tau)\theta'$
Memory capacity	10,000	Replay memory size that contains recent history to update Q-network
Batch size	128	Batch size to sample and use from replay memory when learning
Episode	128	Total episode to learning
Sampling policy	Exponential greedy policy	E-greedy policy that allows agents to explore and exploit

6. Result

The log of the observation of the green team agent is shown in Figure 3. The green team agents generate files with random values, and the more valuable the file, the more time is needed to create. When file generation is completed, the agent will send the file to the central system, the blue team server, and gain a reward. Note that the green agents send the file to the central system, and it must receive the file successfully from the server to earn points. This configuration shows that the natural agent can take independent action with the agent's own internal state, and create a flow that communicates with other agent's processes.



```
green2 - green process: create valuable file /home/worker_green2/greenfile_36_46: reward (+)46

green1 - green process: create valuable file /home/worker_green1/greenfile_36_35: reward (+)35

green4 - green process: create valuable file /home/worker_green4/greenfile_36_32: reward (+)32

blue - blueteam green process: receive reward file from 10.10.13.4 reward - (+)46
```

Figure 3. The log of the observation of the green team agent.

This log value is observation of red team agent as shown in Figure 4. Red team agent communicates with malware and backdoor via CnC attack. Red team deploys malware and periodically notifies red team server that malware is enabled when it is executed.

```
green0 - malware: send alive code

red0 - red team server: get op command 0

red0 - red team server: malware alive from 10.10.13.2
```

Figure 4. The log of the observation of the red team agent.

Upon checking the active status of the malware, the red team launches a phased attack, which indicates that malware deployment status is kept in the internal state and communicates with the processes of other hosts to make a CnC attack (Figure 5).

```
red0 - red team server: get op command 2

red0 - red team server: send list command to 10.10.13.2
```

Figure 5. The log of the red team launches a phased attack.

Red team communicates with compromised hosts via CnC server extracting filenames and instructing them to send specific files. If file is successfully received, the blue team gets negative reward as much as file value (Figure 6).

```
green0 - malware: get command: 4

green0 - malware: send filenames
['/home/worker_green0/greenfile_0_43','/home/worker_green0/greenfile_1_36']



red0 - red team server: get op command 7

red0 - red team server: recive file list from 10.10.13.2

red0 - red team server: get op command 2


green0 - malware: get command: 4

green0 - malware: send filenames ['/home/worker_green0/greenfile_0_43',
'/home/worker_green0/greenfile_1_36']


red0 - red team server: receive file from 10.10.13.2 reward: (-)36
```

Figure 6. The log of the red team communicates with compromised hosts via CnC server.

The blue team takes the status of the hosts and sends the appropriate action commands to the hosts to execute them (Figure 7). This scenario configuration is configured so that all hosts can take action through complex steps with multiple internal states.

```
green0 - blueteam process: process scan

green0 - blueteam process: kill malicious process
```

Figure 7. The log of the blue team takes the status of the hosts.

This scenario requires the defense agent to analyze the complex environment and make the right decision. To evaluate the DQL agent, we compare it with three blue agents—random, sleep, and perfect. Perfect agent is in an ideal situation where there is no red team threat and blue team action in a simulation environment. Therefore, it is the highest value that keeps accumulating points without loss. However, in simulations where there is a threat from the red team, the blue team has to take proper actions that have cost and defend them, so the ideal maximum value cannot be reached. DQL agents can be compared to random agents and sleep agents. Random agent takes random action at every step, so agent can luckily block red team attacks from malware initial access to backdoor installation. However, random action has cost, and reset action interrupts the green team to produce valuable files, resulting in a less final reward. For sleep agents, the blue team takes no counteractions, so it is vulnerable to red team attacks. Therefore, the red team does not respond to the backdoor installation and file leakage after the initial access of malware, and after one attack, the valuable files generated by the green team are continually leaked.

Table 2 shows the characteristics of these agents. It is important that the DQL agent learns the simulation environment well and takes appropriate actions in suspected situations. DQL agent has a very low number of actions taken compared to random agents. Additionally, while the number of actions is low, the number of malware removed is higher, and the number of files leaked to the backdoor is significantly lower due to fast response. When comparing the malware misdetect counted when the detection action is taken even though malware does not exist, the DQL agent is about 20 times the absolute amount and about 2 times the ratio compared to random agent. The result shows DQL agent understands the simulation environment and selects the appropriate action from the explicitly unexposed observation space. Figure 8 shows the total score of each agent. We find that the DQL agent works well compared to other agents, but we can confirm that it is not perfect. We assume that this is caused by a sparse reward problem that can be easily found in reality. There is a way to solve a sparse reward problem like reward shaping, but it is difficult to do so realistically. Therefore, using enhanced algorithms that can solve the sparse reward problem more efficiently will result in performance improvements.

Table 2. Metric for average of 10 episodes. Sleep agent can confirm that the scenario is working well, with 5 penetrations of the backdoor, equal to the number of host computers. Random agent and DQL agent have similar penetrations of malware, but by checking DQL agent's malware remove success rate and scanning misdetect, we can see that DQL agent is trying to act appropriately to a specific state. Additionally, we can see that the backdoor is not removed at this stage and that the DQL agent is trying to respond quickly even if the backdoor is installed.

Metric	DQL Agent	Random	Sleep	Perfect
Score	356,993.3	−1,633,302.9	−1,621,982.8	546,872.9
Action network scan	0.2	2040.2	0.0	0.0
Action scan process	196.3	2080.5	0.0	0.0
Action reset	236.2	2039.5	0.0	0.0
Malware remove	94.4	51.7	0.0	0.0
Malware deploy	302.0	302.1	5.0	0.0
Scanning misdetect	101.9	2028.8	0.0	0.0
Backdoor deploy	185.4	225.3	5.0	0.0
Backdoor exfiltrate file count	132.0	1357.2	13,511.2	0.0

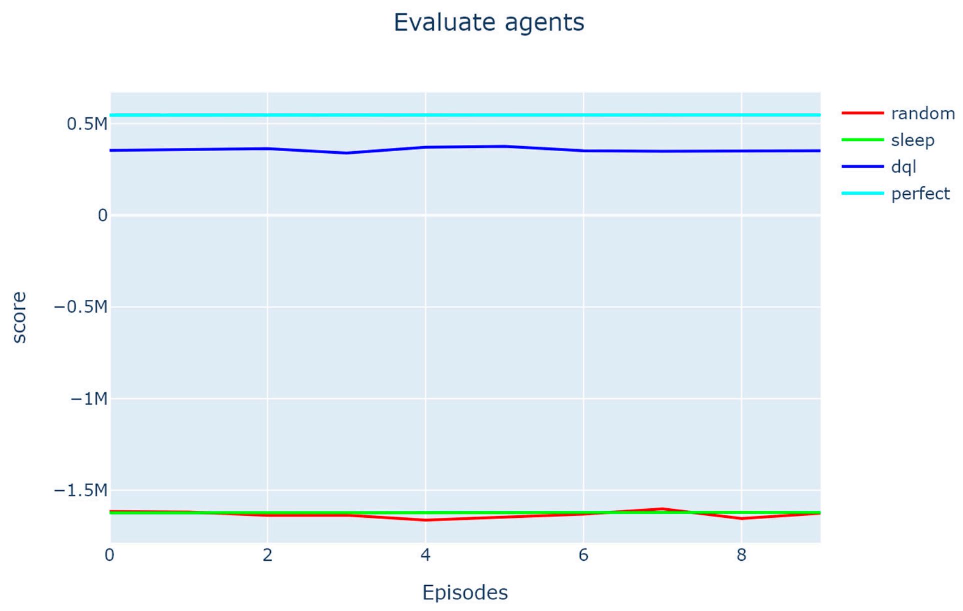


Figure 8. Compare total score of each agent. DQL outperforms sleep and random agents but performs worse than perfect agents. DQL agents can be compared to other agents to confirm that DQL is trained and looking for appropriate behavior, but it is not fully optimal behavior compared to perfect agent.

7. Conclusions

This paper introduces a new cyber environment simulation and learning framework CYE that provides useful components to model complex scenarios. By providing network equipment nodes such as L2switch and routers, CYE can build complex network topologies. The internal state of abstracting the OS and managing it as a process unit helps to flexibly configure various processes and allows for interaction between processes. To show CYE design, we test the framework with a scenario that is inspired by real-world cyber threats by admin@338 group that uses CnC method to show flexible configuration with CYE. The scenario configuration, which uses a CnC server to communicate with malware and undergoes a step-by-step penetration from initial access to backdoor penetration, demonstrates that CYE is capable of flexible and complex configurations that communicate with many agents. With CYE, researchers can easily model complex interactions between nodes and internal states. Additionally, CYE provides useful cloud components like switches and routers to easily configure networking areas. Code-based flexible scenario configuration allows to modeling of flexible environment, observation space, and action.

8. Limitations

For now, CYE only supports generator-based simulations, which can implement time-delayed simulations, but do not fully reflect real-time delays or situations that occur in reality.

9. Future Works

CYE is working in progress. User permission and network modules are implemented in a small range now. Future features of CYE are divided into two main categories. 1. Sophisticated components. Current CYE only provides essential features. Rich components like advanced file systems, networking, etc. will be added. 2. Thread base running. Current CYE environment internal action is run by the generator so that the environment broadcasts step to each node and nodes run internal action by step. A thread-based running environment that enables simulation and more realism remains a challenge.

Author Contributions: M.J.—conceptualization, methodology, software, validation, writing—original draft preparation; J.P.—supervision, validation; S.H.O.—conceptualization, investigation, project administration, funding acquisition. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by a Research Grant from Pukyong National University (2023).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are available upon request.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of this 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. Seh, A.H.; Zarour, M.; Alenezi, M.; Sarkar, A.K.; Agrawal, A.; Kumar, R.; Ahmad Khan, R. Healthcare data breaches: Insights and implications. *Healthcare* **2020**, *8*, 133. [CrossRef]
2. Roberts, L. *The ARPANET Computer Networks, A History of Personal Workstations, A*; ACM: New York, NY, USA, 1986.
3. Sampson, D.; Chowdhury, M.M. The growing security concerns of cloud computing. In Proceedings of the 2021 IEEE International Conference on Electro Information Technology (EIT), Mt. Pleasant, MI, USA, 14–15 May 2021; IEEE: New York, NY, USA, 2021; pp. 50–55.
4. Ensia. 2009. Available online: https://www.enisa.europa.eu/publications/cloud-computing-risk-assessment (accessed on 21 November 2024).
5. Popović, K.; Hocenski, Ž. Cloud computing security issues and challenges. In Proceedings of the 33rd International Convention Mipro, Opatija, Croatia, 24–28 May 2010; IEEE: New York, NY, USA, 2010; pp. 344–349.
6. Ahmad, A.; Hadgkiss, J.; Ruighaver, A.B. Incident response teams–Challenges in supporting the organisational security function. *Comput. Secur.* **2012**, *31*, 643–652. [CrossRef]
7. Williams, P.A.; Woodward, A.J. Cybersecurity vulnerabilities in medical devices: A complex environment and multifaceted problem. *Med. Devices Evid. Res.* **2015**, *8*, 305–316. [CrossRef]
8. McWhorter, D. Mandiant Exposes APT1—One of China's Cyber Espionage Units Releases 3000 Indicators. Mandiant. Available online: https://nsarchive.gwu.edu/document/21484-document-83 (accessed on 18 October 2024).
9. Hawley, S.; Read, B.; Brafman-Kittner, C.; Fraser, N.; Thompson, A.; Rozhansky, Y.; Yashar, S. *Apt39: An Iranian Cyber Espionage Group Focused on Personal Information*; Technical Report; Mandiant: Reston, VA, USA, 2019.
10. Wang, Z.; Zhu, H.; Sun, L. Social engineering in cybersecurity: Effect mechanisms, human vulnerabilities and attack methods. *IEEE Access* **2021**, *9*, 11895–11910. [CrossRef]
11. Hutchins, E.M.; Cloppert, M.J.; Amin, R.M. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Lead. Issues Inf. Warf. Secur. Res.* **2011**, *1*, 80.
12. Arora, M.; Bohrer, S.; Yoachimik, O.; Doucette, C.; Forster, A.; Wood, N. How Cloudflare Auto-Mitigated World Record 3.8 Tbps Ddos Attack. 2024. Available online: https://blog.cloudflare.com/how-cloudflare-auto-mitigated-world-record-3-8-tbps-ddos-attack (accessed on 13 February 2025).
13. He, Z.; Zhang, T.; Lee, R.B. Machine learning based DDoS attack detection from source side in cloud. In Proceedings of the 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), New York, NY, USA, 26–28 June 2017; IEEE: New York, NY, USA, 2017; pp. 114–120.
14. Aldhyani, T.H.; Alkahtani, H. Attacks to automatous vehicles: A deep learning algorithm for cybersecurity. *Sensors* **2022**, *22*, 360. [CrossRef] [PubMed]
15. Wang, W.; Guo, J.; Wang, Z.; Wang, H.; Cheng, J.; Wang, C.; Yuan, M.; Kurths, J.G.; Luo, X.; Gao, Y. Abnormal flow detection in industrial control network based on deep reinforcement learning. *Appl. Math. Comput.* **2021**, *409*, 126379. [CrossRef]
16. Microsoft Defender Research Team; Seifert, C.; Betser, M.; Blum, W.; Bono, J.; Farris, K.; Goren, E.; Grana, J.; Holsheimer, K.; Marken, B.; et al. CyberBattleSim. 2021. Available online: https://github.com/microsoft/cyberbattlesim (accessed on 2 November 2024).
17. Molina-Markham, A.; Miniter, C.; Powell, B.; Ridley, A. Network environment design for autonomous cyberdefense. *arXiv* **2021**, arXiv:2103.07583.
18. Baillie, C.; Standen, M.; Schwartz, J.; Docking, M.; Bowman, D.; Kim, J. Cyborg: An autonomous cyber operations research gym. *arXiv* **2020**, arXiv:2002.10667.

19. Janisch, J.; Pevný, T.; Lisý, V. Nasimemu: Network attack simulator emulator for training agents generalizing to novel scenarios. In *European Symposium on Research in Computer Security*; Springer Nature: Cham, Switzerland, 2023; pp. 589–608.
20. Nguyen, H.P.T.; Chen, Z.; Hasegawa, K.; Fukushima, K.; Beuran, R. PenGym: Pentesting Training Framework for Reinforcement Learning Agents. In Proceedings of the 10th International Conference on Information Systems Security and Privacy (ICISSP 2024), Roma, Italy, 26–28 February 2024; pp. 498–509.
21. FT Intelligence, China-Based Cyber Threat Group Uses Dropbox for Malware Communications and Targets Hong Kong Media Outlets. 2015. Available online: https://cloud.google.com/blog/topics/threat-intelligence/china-based-threat/?hl=en (accessed on 8 November 2024).
22. Strom, B.E.; Applebaum, A.; Miller, D.P.; Nickels, K.C.; Pennington, A.G.; Thomas, C.B. *Mitre Attck: Design and Philosophy*; Technical Report; The MITRE Corporation: McLean, VA, USA, 2018.