Evaluation and Comparison of the Use of Reinforcement Learning Algorithms on SSH Honeypot

Marco Ariano Kristyanto[1], Maya Hilda Lestari Louk[2]

[1,2]Informatics Department, University of Surabaya, Surabaya, East Java, Indonesia
Email: [1*]marcokristyanto@staff.ubaya.ac.id, [2]mayalouk@staff.ubaya.ac.id

(Received: 9 Jan 2024, revised: 31 Jan 2024, accepted: 2 Feb 2024)

Abstract

A honeypot is a tool or system used to record, redirect, and even lure hackers into penetrating and exploiting a system. The increasing development of technology causes cyber hackers to realize the existence of honeypots using various other software and tools. So, honeypots need a way to learn how hackers behave. The idea proposed is to combine honeypots with reinforcement learning algorithms so that honeypots become adaptive honeypots. This study suggests the concept by comparing the two Q learning-based RL algorithms, namely DQN and DDQN, to reach which algorithm is more optimal. The study results showed that the DDQN algorithm is more optimal in determining actions when compared to the DQN algorithm because using a double Q-value can help determine the action more accurately. Based on the result, the DDQN algorithm consumed less memory than the DQN Honeypot. The learning rate curve and the processing of DDQN algorithm commands can be used as an alternative algorithm that can be combined with honeypots because of the learning rate, which can make honeypots faster in the dynamic environment.

Keywords: Honeypot, Reinforcement Learning, DDQN, DQN, Adaptive Honeypot

I. INTRODUCTION

The development of technology has also led to an increase in the threat of cybercrime. One way to deal with cybercrime is to implement a honeypot as an intrusion detection system. A Honeypot is a tool for luring the attacker to observe and analyze their method [1]. Honeypots can be used as intrusion detection systems to monitor and respond to computer abuse. A Honeypot is a Network Intrusion Detection System that detects and writes attackers' activity or counterattack [2].

Traditionally, a honeypot has been used to capture and study malware activities for a long time. The Server-side honeypots are designed to intercept assaults and gather malicious requests and malware by mimicking software or services susceptible to compromise. This facilitates the development of intrusion detection systems, understanding network and web attacks, and stopping spam emails.[3].

However, as technology advances and attackers become more sophisticated, new problems have arisen; many attackers know how to detect the honeypot and how to evade it. Many honeypots spread worldwide and are statically configured, which means they do not see the change in attacker behavior [4]. New methods are needed to combat this issue. One of the ideas is the making of an adaptive honeypot. The idea proposed by researchers is how to make a honeypot that can learn and interact with attacker behavior [5], which allows for more extended interactions between the attacker and the honeypot.

Researchers combine the honeypot with a reinforcement learning algorithm to make a honeypot that can learn from the attacker's behavior. Despite being both supervised and unsupervised, reinforcement learning (RL) is one area of machine learning. This algorithm is used to interact with the dynamic situation. In cyber security, RL is used to construct the attack mitigation scheme [6]. Sethi et al. [7] define the components of the RL algorithm as agent, reward, state, action, policy, and value.

Many studies related to adaptive honeypots, some of which are as follows. Pauna and Bica [8] merged the SSH Honeypot Kippo and RL algorithm to create RASSH, which used the SARSA algorithm with the Markov decision process and developed it using the Pybrain library. The research continued by Pauna et al. [9] when they merged the Cowrie Honeypot with the DQN Algorithm to create QRASSH. To prevent attackers from using the honeypot detection tools, Suratkar et al. [10] joined the Cowrie honeypot with the DQN Algorithm so it could decoy the honeypot detection tools that attackers use.

As stated above, the contribution of these papers is as follows:

- To determine the performance of the RL Algorithm used in adaptive honeypots, especially the learning rate, in this research, researchers compare DQN and DDQN algorithms.
- To examine the DDQN algorithm performance when merged with the honeypot.
- Add more measurement metrics for deeper analysis, like the impact of environment hardware specification on adaptive honeypot.

The structure of the remaining papers is as follows: Section I will present the background of the research, section II explain the literature review, and Section III discuss the related works used in this research. Section IV will describe the system architecture used in the study, Section V discuss the experimental results, and Section VI discuss the conclusion and future works.

II. LITERATURE SURVEY

a. Honeypot

Dowling et al. [11], in these articles, state that honeypots were first introduced in 1993, have evolved, and are now used to match the emerging threat. Touch & Colin [12] state that a honeypot is a system used to decoy and trap attackers when they enter the system. Based on the level of interaction, Zemene and Avadhani [13] categorized honeypot as follows :

- Low-level interaction honeypot
  Honeypots that simulate a single service from a real server are low-level interaction honeypots. For example: Kippo, Cowrie.
- High-level interaction honeypot
  Honeypots, such as Honeynet, need a complete system to operate as a server.

The primary objective of honeypot deployment is to trace the infection process, detect malware, and inspect the entire picture of botnet activities from the viewpoint of infected hosts [14]. Honeypots play a crucial role in capturing and analyzing malware and malicious activity. The honeypot collects information, including IP addresses, commands entered into the system, and timestamps [5].

Honeypots are used for many cybersecurity purposes, such as classifying SSH bruteforce attacks using machine learning [15]. This research used the Kippo honeypot to deal with brute force attacks and then used machine learning classification from the honeypot log to find the best classification techniques.

Bellekens et al. [16] explain that honeypots and other interactive defenses are frequently built on actual or virtualized systems that record harmful users' inquiries and interactions. Although honeypots help spot and thwart cyberattacks, their deceitful function is commonly overlooked.

b. Cowrie

Cowrie is a honeypot that lies between SSH and the telnet band. It is specifically developed for recording brute-force attacks and the shell interaction performed by the adversary. When in high interaction mode (proxy), it acts as an SSH and telnet proxy to watch attacker activity on another system. In medium interaction mode (shell), it simulates a UNIX system in Python [17].

The development of cowrie uses the Kippo honeypot as the base structure and feature. The purpose of Kippo is to capture enemy shell interactions and brute force attacks. With Kippo, one can create and remove files using a fictitious file system that resembles Debian Linux.

As mentioned before, there is much research conducted about honeypots because, with the growth of cyber-attacks, many attackers can detect honeypot presence using specific tools. In the below section, we will discuss the adaptive honeypot, one of the main topics in this article.

c. Adaptive Honeypot

Shi et al. [14] discuss in this research that the globally distributed honeypots are statically configured. Dowling et al. [11] also mentioned that as soon as malware developers learned about honeypots, they began recording and examining assault activities. To overcome and solve this problem, researchers proposed the concept of an adaptive honeypot, a honeypot that can learn from adversaries and amend their actions based on reinforcement learning [4].

The other research about an adaptive honeypot is to make honeypots more aggressive when dealing with attackers. Djanali et al. [18] conducted the aggressive honeypot to deal with XSS attacks and SQL Injection. The method that they proposed is a honeypot that can open the identity of the attacker when accessing the honeypot by sending the malicious code using likejacking techniques.

Pauna and Bica [8] combined the Kippo Honeypot with SARSA Algorithm and named it RASSH, which it developed using the my brain library. The study of adaptive honeypots, continued by Pauna et al. [9], uses the RL algorithm Q Learning combined with Cowrie Honeypot to decide how to interact with attackers.

Using an adaptive honeypot as a cyber defense tool is one of the exciting areas in cyber security. The combination of the honeypot and the machine learning algorithm is determined to achieve two different goals: to engage with the attacker and to guard against the risk of being compromised [19]. The other benefit of using an adaptive honeypot is that it can covert against widely used survey and honeypot detection tools [20].

d. Reinforcement Learning

Reinforcement Learning (RL) is a branch of machine learning despite supervised and unsupervised learning [21]. The components of RL, as mentioned by Gupta and Katarya [22], are:

a. Agent: an agent is a system that does some action to obtain the reward.
b. State: The action dictates the agent's current condition and any modifications.
c. Action: A behavior from the system.
d. Environment: a place where the agent operates or moves.

e. Reward: Beneficial result of the agent's action
f. Discount factor: it is a factor of the future reward calculated by the agent
g. Penalty: The rules that the agent follows until the next stage. The policy is based on the user's current stage.

The mathematical models from reinforcement learning can be described in formula 1 as follows :

$$V(s) = \sum_t \gamma^t (S_t) \quad (1)$$

Note:
V(s) represent the function future reward from future states.
γ^t represents the discount factor.
St represents the state.

Nowadays, RL algorithms are used in many aspects; despite being used in artificial intelligence, they are employed in cyber security. Deep Reinforcement Learning (DRL) in cyber security can solve complex and high-dimensional cyber defense problems [23]. Using the RL algorithm to determine how many honeypots are to be deployed to protect the actual entities, the result is that both Q-learning and epsilon greedy demonstrate the efficacy of both methods [24].

e. Q-learning

Q-learning is one of the RL algorithms that is used in many aspects. This algorithm aims to calculate the state action value [9]. Suwannalai and Porpasert [25] state the purpose of this algorithm is to maximize the cumulative discounted reward of the Bellman formula 2 as follows:

$$\mathrm{Q(St, At)} = \mathrm{E}[\, R_{t+1} + 1 + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + |\mathrm{St, At}|\mathrm{I} \quad (2)$$

Where :
α is a learning rate variable filled with parameter values between 0 and 1 and determines the speed of learning ability.
γ is a discount factor filled with parameters between 0 and 1 and determines how the future reward is used to replace the value of the Q-value.

Q-Value / state-action value is a variable used to calculate the value of the pair of state and action, where this value will later be used to find the maximum value before determining the magnitude of the action.

Tageson [26] explains Q-learning uses a table to store the values of all actions for all states. In each iteration, the agent looks up the table's highest-valued action for a given state.

Q-learning is a model-free approach that updates the Q-values estimation based on the experience samples on each time step [27].

f. DQN

DQN is a combination of Q-learning and neural networks. In the DQN algorithm, states are determined through input to the neural network, which calculates the Q value of all actions. This algorithm aims to select the maximum value that depends on the full reward [28]. The two critical parameters of the DQN algorithm are the Target network and Experience Replay. Suwannalai and Polpasert [25] explain that Experience Replay in DQN is a repetition of the previous process.

Much research combines the DQN algorithm with cybersecurity tools like honeypot and IDS because the policy network is simple and fast, which is suitable for online learning and rapid responses in modern data networks with evolving environments, especially in cyber security [23], [29].

g. DDQN

DDQN is an RL algorithm that is formed from the modification of DQN. The DDQN algorithm was first introduced by Van Hasselt [30]. He explained that the weakness of Q-learning is that calculating the Q value can make a biased result because the overestimation leads to a limited value. The result value is not the optimal value. Therefore, as a further adaptation from the Q-Learn algorithm, two Q-val calculations aim to obtain maximum results in formula 3 as follows:

$$Q^*(s_t, a_t) \approx r_t + \gamma Q_\theta\ (s_{t+1}, argmax_{a'} Q_{\theta'}(s_{t+1}, a')) \quad (3)$$

Where:
Q* is the Q Value of the Model
Q' is the Q Value of the Target Model.

In cyber security, many researchers also use this algorithm to combine it with IDS, and the result is the performance of this algorithm in IDS outperforms the other algorithms, especially in learning and performance [27].

This research uses the DQN and DDQN algorithms from the previous section because the ability to handle temporal space and the simple policy network can be applied to determine the dynamic environment, especially in a honeypot environment.

III. RELATED WORKS

Much research about adaptive honeypots has been conducted; this section will discuss the related works in adaptive honeypots and similar studies about the research. As mentioned before, the concept of adaptive honeypot was proposed by researchers because the honeypots that are distributed around the world are statically configured, which requires prior knowledge about the attacker [4]. Shi et al. [14] conducted the model of a mimicry honeypot, which could perceive and adapt to the change of the network service and perform better camouflage.

Pauna and Bica [8] developed the RASSH honeypot; they combined it with the SARSA Algorithm and can interact with the attacker. In 2018, Pauna et al. introduced the QRASSH, which is the combination of SSH Honeypot Cowrie with RL Algorithm Deep Q Network; the result of the research is an adaptive honeypot that can decide how to interact with the attacker. Dowling et al. [10] use reinforcement learning to conceal the honeypot's ability, and from the research, deploying adaptive honeypots can make prolonged

interaction. It can cause the honeypot's presence undetected by the attacker.

Deploying an adaptive honeypot can help predict future attack patterns because the adaptive honeypot interaction can create a more realistic interaction with the attacker [28]. Suratkar et al. [9] state that deploying adaptive honeypots, with some severity analysis, can help deal with honeypot detection tools. Marco et al. [31] discuss using the Q network algorithm and compare it with the unmodified honeypot; the result is the interaction time from the adaptive honeypot is longer than the nonadaptive honeypot.

The honeypot is one of the most common approaches to protecting the network against attacks, such as MITM (Man in the Middle) and DDOS attacks [32]. Their work suggests a Markov Decision Process (MDP) known as the state-action-reward-state-action (SARSA) for honeypot design. Compared to conventional IDSs, the suggested system that uses environmental trials can achieve higher accuracy and faster convergence.

IV. SYSTEM ARCHITECTURE

This section will discuss the system architecture used in this research. This section will discuss the honeypot structure, the RL module, the honeypot installation, and the measurement of the RL Algorithm used in this research.

a. Honeypot Architecture

The honeypot that is used in this research is a Cowrie-based honeypot. In Cowrie source code, an Action Module has been implemented to allow console interaction with the attacker. The adaptive honeypot has been deployed and implemented in Alibaba Cloud's simple application server.

The honeypot's structure is determined using the QRASSH base honeypot model [9]. It uses Q-learning to increase the interaction between the honeypot and the attacker. In this honeypot, the action module determines what action to take using the Q learning calculation. The actions implemented in this research are as follows:

- Allow: permit the command
- Block: block the execution of Linux command in a console
- Delay: delay the execution of Linux commands in a console
- Fake: change the output of the command by substituting the command.
- Insult: it modifies the command output by showing the insult message in a language according to the IP address.

The explanation of the workflow from Figure 1 can be described as follows. After the honeypot is started, it receives commands that an attacker inputs; after that, the RL module will determine the action based on the RL algorithm injected in the honeypot system; if the blocked action has been chosen, the system will be stopped. In the RL module, the process of the Q learning calculation is determined in the module.

From Figure 1, the process of the Q-learning algorithm can explained as follows. For Step 1, initialize the Q value before receiving the input. After obtaining the input, the next step is to choose an action based on the Q calculation. After the action has been selected, the next step is to measure the reward R, and the final step is to update the Q values and go back to step 2 again. The process of the Q-learning algorithm is the same as the process in the research from Pauna et al. [9].

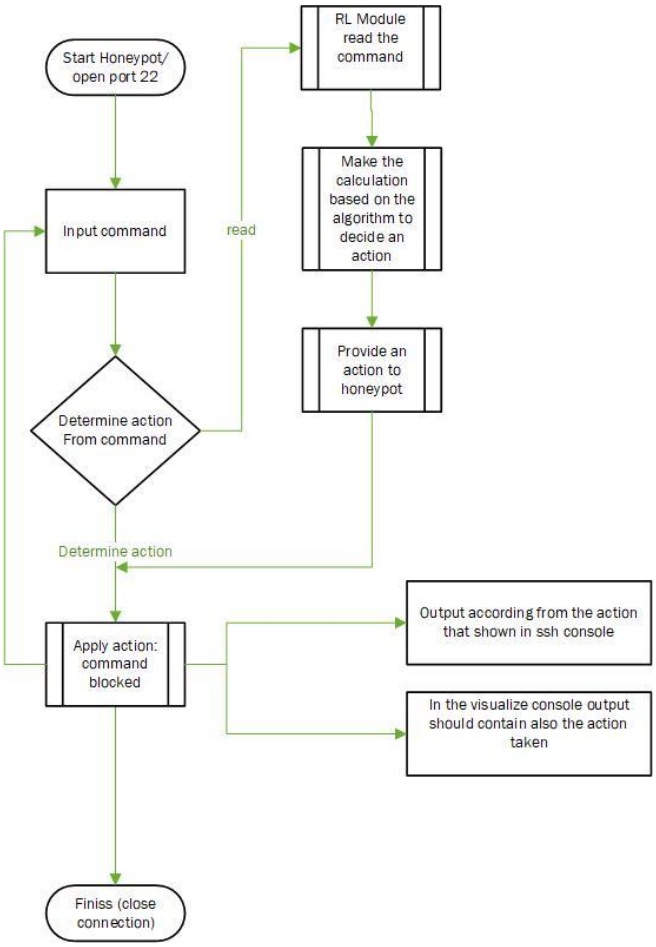

Figure 1. Adaptive Honeypot Diagram

Reinforcement learning is employed here to determine the best answer to every command to prevent the attacker from ending the session. Initially, The honeypot will respond unpredictably (epsilon-greedy), and when more assaults occur, it will discover the ideal values to Complete the Q-table and act based on the results.

b. Honeypot Analysis

Because of the RL Algorithm, this research will not use labeled class and ground truth for the honeypot analysis. In this research, the analysis that we measured as follows:

Total command executed by an attacker, Learning rate of the RL Algorithm, and Command mapping: the simulation of the attack that adaptive honeypot can handle.

In this research, the attacker command will categorised as follows:

- Checkconf: Command used to check configuration to the system (whoami, ps, uptime, top, ifconfig, name, history, id)
- Download: Command used to download the remote files (wget, FTP, curl)
- Install: Command used to install the new software (tar, unzip, cp, mkdir, chmod, mv)
- Changeconf: Command used to change the configuration of the system (kill, useradd, nano, sshd, userdell, vim)
- Passwd: Command used to modify the password of the user (passwd)
- Run: Command used to run a program (./).

c. Reward Module Calculation

The reward module calculation aims to maximize the interaction between the honeypot and the attacker. This reward function works because the attacker executes the download command. More information about the reward function is described as follows:

- If the attacker runs the download command, it will get a reward of 500
- If the check conf command is executed, it will get a reward of 200
- Meanwhile, if the Linux command that has been implemented is executed but not a download command, it will get 0 rewards.
- If the Linux command has not been implemented and executed but is not a download command, it will get a -200 reward.
- If the attacker decides to close the connection, it will get a -500 reward.

V. RESULT AND DISCUSSION

This section will discuss the result of deploying and installing a honeypot in the Alibaba cloud. Moreover, this section will present the honeypot analysis and the measure of the learning rate algorithm of the adaptive honeypot. Honeypot was installed in two virtual machines in Alibaba Cloud, and both machines ran from December 1, 2023, until January 1, 2024.

Both machines sent the output into the kippo graph to display the activity in two honeypots. Also, both honeypots have been set up to send the output into MySQL.

a. Comparison between DQN and DDQN Honeypot

This section will compare command transitions in the DDQN Honeypot and DQN Honeypot using QRASSH as the base honeypot. In Table 1, we will see the comparison between DQN and DDQN honeypots during the processing of the attack command transition.

In Figure 2 and Figure 3, there is the number of connections based on the IP Address and country of origin attackers.

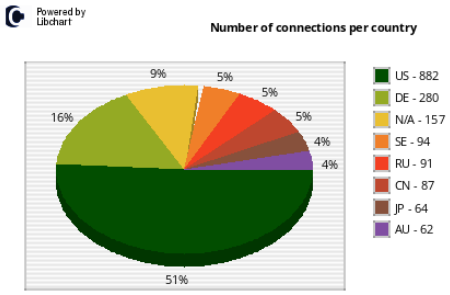

Figure 2. Connection based on IP Address and Country in DQN Honeypot

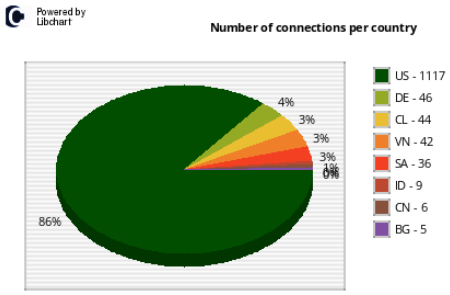

Figure 3. Graphic Connection based on IP Address and Country in DDQN Honeypot

In Figure 2 and Figure 3, the most connection attempts by attackers in two honeypots are from the USA. In DQN Cowrie, the total connection from the USA is 880 connections; in the DDQN honeypot, the most connections from the USA are 1,117 connections; in both honeypots, the 2nd most successful IP address is from Germany.

From both figures, the 2nd IP the attacker used to interact with the honeypot is from Danish. It can be stated that most attackers who interact with both honeypots are from the USA and Denmark. The other countries , are spread in various countries, like Vietnam, Russia, China, etc.

The study also looked at how the two activities in the honeypot were compared during that period, and the results of the activities in each honeypot are shown in Figure 4 and Figure 5 below. In both figures, it can be seen that in each honeypot, there was an increase in attack activity from December 28, 2023, to January 1, 2024.

The increase was very significant, especially in DDQN Honeypot, where the number of incoming attacks was relatively high and rose drastically in that period. From both diagrams, these two adaptive honeypots work well in luring attackers to access and execute some commands.

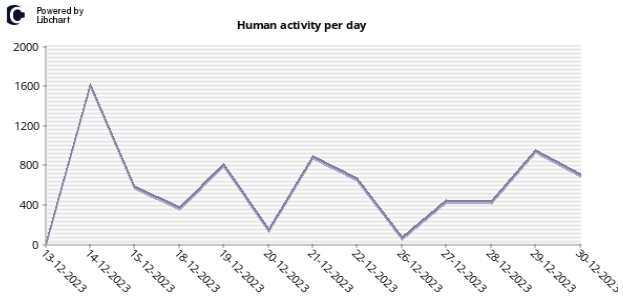

Figure 4. Human Activity in DQN Honeypot

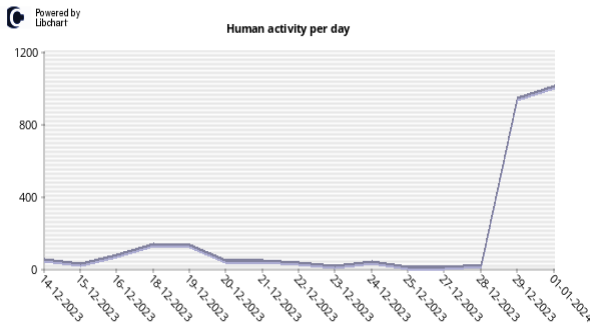

Figure 5. Human Activity in DDQN Honeypot

In addition to comparing the activity of the two types of honeypots, the study also checked the username combinations used by attackers to access the honeypot directly. Most combinations of usernames used can be seen in Table 2 and Table 3

Table 1. Table of command Transitions that Attackers Input on Both Honeypot Types

No	Command	Action	Honeypot DQN	Honeypot DDQN
1	Df	Allow	5	3
		Block	2	1
		Insult	2	4
2	Ifconfig	Allow	1	2
		Block	3	1
		Insult	2	1
3	Chmod	Allow	575	0
		Block	586	1
		Insult	516	0
4	Echo	Allow	547	4
		Block	584	6
		Insult	555	5
5	Mkdir	Allow	538	0
		Block	585	1
		Insult	544	2

Table 2. Combination of Username and Password Used to Access the DDQN Honeypot

No	Username	Password	Total
1	Root	Broadguam1	140
2	Admin	Admin	42
3	Root	123456	23
4	Root	1234	20
5	Root	12345678	16

Table 3. Combination of Username and Password Used to Access the DQN Honeypot

No	Username	Password	Total
1	root	3245gs5662d34	2519
2	345gs5662d34	345gs5662d34	2504
3	root	broadguam1	73
4	xg	xg1234	41
5	Intell	123	41

In Table 2 and Table 3, there is a similarity in the combination of username and password used to enter the two honeypots, which is a combination of root username and broadguam1 password, wherein the DDQN honeypot, there are 140 attempts. The DQN has 73 attempts to access using the username and password combination.

Furthermore, the study also measured the ability of the RL learning algorithm to adapt to changes in action that occurred in each state. In this study, both used the Adam optimizer and a learning rate of 0.001 to manufacture artificial neural networks. From the results carried out for the duration of the study, results were obtained in Figure 6 and Figure 7 as follows.

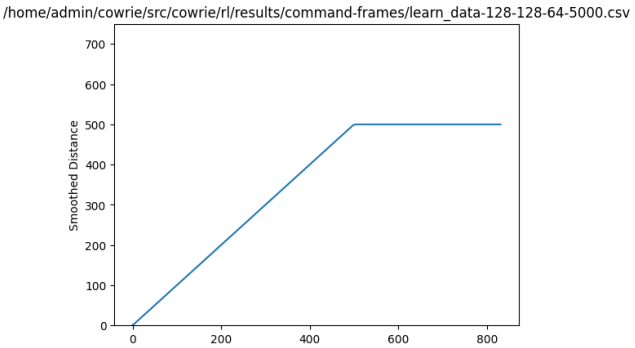

Figure 6. Learning Rate Graphic Honeypot DQN

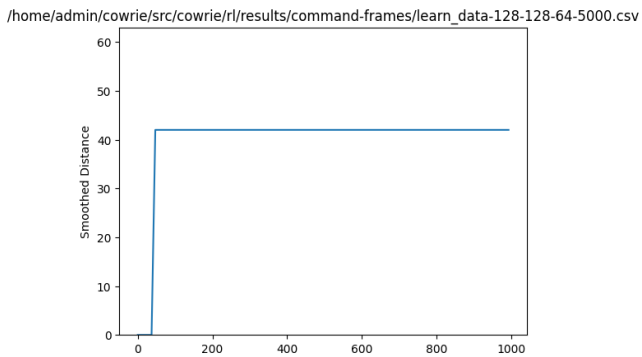

Figure 7. Grafik Learning Rate Graphic Honeypot DDQN

Table 4. The Comparison of Algorithm Result

Parameters	DQN	DDQN
Episode	1000	1000
Step of smoothed	50	500

DOI: 10.34148/teknika.v13i1.763

distance learning		
Memory usage	756 MB/2 GB	800 MB/2 GB
The total download command executed	25	45

Through Figures 6 and 7, it can be seen that DDQN Honeypot has a better learning rate ability compared to DQN Honeypot. It can be noticed that in DDQN Honeypot, the learning curve increases in the smoothed distance at 50 and then stabilizes to step 1,000, while in DQN Honeypot learning rate curve, the learning rate curve is only stable at 500 when entering epoch 500 and above. This indicates that the learning ability of the DDQN algorithm, when in discrete conditions, can more quickly adapt to changes in state conditions because it uses calculations of 2 times the value of Q to determine the correct action used to respond to state changes.

An explanation of the algorithm comparison result can be found in Table 4. Table 4 explains the result of comparing the combination of DQN and DDQN with the Cowrie honeypot. The usage memory between the two algorithms is shown in Table 4, demonstrating that the DDQN honeypot usage is less than that of the DQN Honeypot. More download commands are executed in DDQN Honeypot than in DQN Honeypot.

In Table 1 of the two honeypots in response to the commands entered and determining the selected action based on the calculation of the Q matrix, sampling was taken of 5 commands entered, and the results were obtained that for the system exploitation command, namely ifconfig, in DDQN Honeypot the most selected action allowed alias let the command run, while in DQN Honeypot, The most actions taken are blocks.

Then, for the mkdir command on DQN Honeypot, the most actions taken are blocks, while on DDQN Honeypot, the most actions are insults. From the results of this observation, the DDQN honeypot can be said to be more optimal in taking action because it goes through a process of twice calculating the Q value, as explained in the previous chapter.

Observations were also made to observe what commands were carried out by each honeypot during the study period, and are shown in Figure 8 and Figure 9 below.

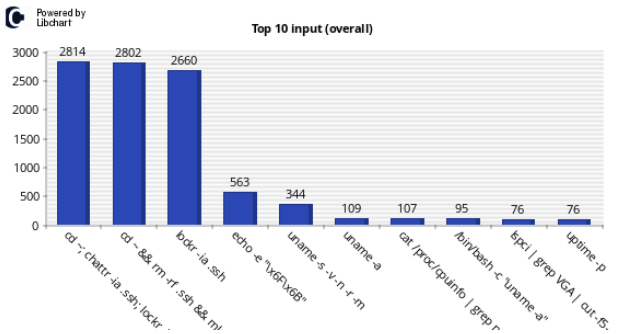

Figure 8. Command Graph Recorded by DQN Honeypot

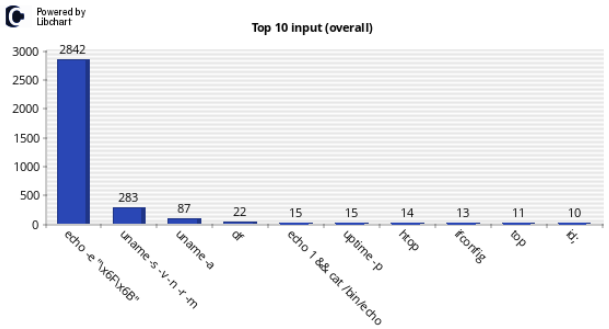

Figure 9. Command Graph Recorded by DDQN Honeypot

Figure 8 shows that most commands executed in the DQN Honeypot are commanded to modify the system, namely doing the chatter (change attribute) command and the rm (remove) directory command. SSH.

Meanwhile, in the DDQN Honeypot in Figure 9, it is obtained that the command executed by the attacker is an echo command with hexadecimal characters, indicating that the user is making sure they are interacting with the honeypot or the original system because the attacker also sees how long the system uptime and what usernames are running.

VI. CONCLUSION AND FUTURE WORK

This section will discuss the conclusion and future work from this research. About the benefit of using an adaptive honeypot to improve the network defense, which RL algorithm has a better result, and the future improvement from the future research.

a. Conclusion

From this research, it was concluded that the DDQN algorithm can be integrated with SSH Honeypot. DDQN algorithm is used to determine each action more optimally because it uses 2 Q-Values calculation; this can be seen in response to specific commands. DDQN can help the honeypot learn faster than DQN, it can be noticed that the DDQN honeypot uses the memory server less than the DQN honeypot.

DDQN algorithm can be applied as an alternative algorithm to be merged with the honeypot. Because the usage memory is less than that of the DQN algorithm. In addition, the graph of transitions between commands also shows that for some commands, DDQN has more accurate predictions when compared to the same DQN algorithm combined with similar honeypots. It complements the previous research, that explained the combination of DDQN algorithm with IDS.

b. Future work

In future work, there is more scope for future research, for example, in the reward module, which is used by complex code for the reward module. There is room for inverse reinforcement learning to find the optimal reward function to increase the attackers' engagement.

Other RL optimization techniques, Such as PPO and A2C, are also possible, which can show promise in situations with more attacks. Further work can be done on the honeypot forensics to utilize the data obtained better, making the backtracking process easier.

For the next improvement suggestion, there is also some honeypot data severity analysis to analyze the pattern of command input by attackers because the dataset collected from honeypots can obtain more helpful information than unmodified honeypots.

REFERENCE

[1] R. Vishwakarma and A. K. Jain, "A honeypot with machine learning based detection framework for defending IoT based botnet DDoS attacks," *Proc. Int. Conf. Trends Electron. Informatics, ICOEI 2019*, no. Icoei, pp. 1019–1024, 2019, doi: 10.1109/ICOEI.2019.8862720.

[2] F. Zhang and S. cheng Khoo, "An empirical study on clone consistency prediction based on machine learning," *Inf. Softw. Technol.*, vol. 136, Aug. 2021.

[3] C. Yang, J. Zhang, and G. Gu, "A taste of tweets: Reverse engineering twitter spammers," *ACM Int. Conf. Proceeding Ser.*, vol. 2014-Decem, no. December, pp. 86–95, Dec. 2014, doi: 10.1145/2664243.2664258.

[4] G. Wagener, R. State, T. Engel, and A. Dulaunoy, "Adaptive and self-configurable honeypots," *Proc. 12th IFIP/IEEE Int. Symp. Integr. Netw. Manag. IM 2011*, pp. 345–352, 2011, doi: 10.1109/INM.2011.5990710.

[5] D. Fraunholz, M. Zimmermann, and H. D. Schotten, "An adaptive honeypot configuration, deployment and maintenance strategy," *Int. Conf. Adv. Commun. Technol. ICACT*, pp. 53–57, Mar. 2017, doi: 10.23919/ICACT.2017.7890056.

[6] J. Wang, J. Liu, H. Guo, and B. Mao, "Deep Reinforcement Learning for Securing Software-Defined Industrial Networks With Distributed Control Plane," *IEEE Trans. Ind. Informatics*, vol. 18, no. 6, pp. 4275–4285, 2022, doi: 10.1109/TII.2021.3128581.

[7] K. Sethi, Y. V. Madhav, R. Kumar, and P. Bera, "Attention based multi-agent intrusion detection systems using reinforcement learning," *J. Inf. Secur. Appl.*, vol. 61, p. 102923, Sep. 2021, doi: 10.1016/J.JISA.2021.102923.

[8] A. Pauna and I. Bica, "RASSH - Reinforced adaptive SSH honeypot," *IEEE Int. Conf. Commun.*, 2014, doi: 10.1109/ICCOMM.2014.6866707.

[9] A. Pauna, A.-C. Iacob, and I. Bica, "QRASSH - A Self-Adaptive SSH Honeypot Driven by Q-Learning," pp. 441–446, Oct. 2018, doi: 10.1109/ICCOMM.2018.8484261.

[10] S. Suratkar *et al.*, "An adaptive honeypot using Q-Learning with severity analyzer," *J. Ambient Intell. Humaniz. Comput.*, vol. 13, no. 10, pp. 4865–4876, Oct. 2022, doi: 10.1007/S12652-021-03229-2/TABLES/7.

[11] S. Dowling, M. Schukat, and E. Barrett, "Using reinforcement learning to conceal honeypot functionality," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11053 LNAI, pp. 341–355, 2019, doi: 10.1007/978-3-030-10997-4_21/COVER.

[12] S. Touch and J.-N. Colin, "A Comparison of an Adaptive Self-Guarded Honeypot with Conventional Honeypots," *Appl. Sci.*, vol. 12, no. 10, p. 5224, May 2022, doi: 10.3390/APP12105224.

[13] M. S. Zemene and P. S. Avadhani, "Implementing high interaction honeypot to study SSH attacks," *2015 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2015*, pp. 1898–1903, Sep. 2015, doi: 10.1109/ICACCI.2015.7275895.

[14] A. Shimoda, T. Mori, and S. Goto, "Sensor in the dark: Building untraceable large-scale honeypots using virtualization technologies," *Proc. - 2010 10th Annu. Int. Symp. Appl. Internet, SAINT 2010*, pp. 22–30, 2010, doi: 10.1109/SAINT.2010.42.

[15] M. A. Kristyanto *et al.*, "SSH Bruteforce Attack Classification using Machine Learning," *2022 10th Int. Conf. Inf. Commun. Technol. ICoICT 2022*, pp. 116–119, 2022, doi: 10.1109/ICOICT55009.2022.9914864.

[16] X. Bellekens *et al.*, "From Cyber-Security Deception to Manipulation and Gratification Through Gamification," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11594 LNCS, pp. 99–114, 2019, doi: 10.1007/978-3-030-22351-9_7/FIGURES/9.

[17] M. Oosterhof, "Cowrie Documentation," p. 1, 2022, [Online]. Available: https://readthedocs.org/projects/cowrie/downloads/pdf/latest/

[18] S. Djanali, F. Arunanto, B. A. Pratomo, A. Baihaqi, H. Studiawan, and A. M. Shiddiqi, "Aggressive web application honeypot for exposing attacker's identity," *2014 1st Int. Conf. Inf. Technol. Comput. Electr. Eng. Green Technol. Its Appl. a Better Futur. ICITACEE 2014 - Proc.*, pp. 212–216, Mar. 2015, doi: 10.1109/ICITACEE.2014.7065744.

[19] S. Touch and J. N. Colin, "Asguard: Adaptive Self-guarded Honeypot," *Int. Conf. Web Inf. Syst. Technol. WEBIST - Proc.*, vol. 2021-Octob, no. Webist, pp. 565–574, 2021, doi: 10.5220/0010719100003058.

[20] C. Guan, H. Liu, G. Cao, S. Zhu, and T. La Porta, "HoneyIoT: Adaptive High-Interaction Honeypot for IoT Devices Through Reinforcement Learning," *WiSec 2023 - Proc. 16th ACM Conf. Secur. Priv. Wirel. Mob. Networks*, vol. 11, pp. 49–59, May 2023, doi: 10.1145/3558482.3590195.

[21] R. S. Sutton and A. G. Barto, "Reinforcement learning: An Introduction Second edition," *Learning*, vol. 3, no. 9, p. 322, 2012.

[22] G. Gupta and R. Katarya, "A Study of Deep Reinforcement Learning Based Recommender Systems," *ICSCCC 2021 - Int. Conf. Secur. Cyber Comput. Commun.*, pp. 218–220, May 2021, doi: 10.1109/ICSCCC51823.2021.9478178.

[23] T. T. Nguyen and V. J. Reddi, "Deep Reinforcement Learning for Cyber Security," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 34, no. 8, pp. 3779–3795, 2023, doi: 10.1109/TNNLS.2021.3121870.

[24] P. Radoglou-Grammatikis *et al.*, "Strategic Honeypot Deployment in Ultra-Dense Beyond 5G Networks: A Reinforcement Learning Approach," *IEEE Trans. Emerg. Top. Comput.*, 2022, doi: 10.1109/TETC.2022.3184112.

[25] E. Suwannalai and C. Polprasert, "Network Intrusion Detection Systems Using Adversarial Reinforcement Learning with Deep Q-network," in *International Conference on ICT and Knowledge Engineering*, 2020, vol. 2020-November. doi: 10.1109/ICTKE50349.2020.9289884.

[26] D. Tagesson, N. Xiong, and S. Barua, "a Comparison Between Deep Q-Learning and Deep Deterministic Policy Gradient for an Autonomous Drone in a Simulated Environment," 2021.

[27] H. Alavizadeh, H. Alavizadeh, and J. Jang-Jaccard, "Deep Q-Learning Based Reinforcement Learning Approach for Network Intrusion Detection," *Computers*, vol. 11, no. 3, 2022, doi: 10.3390/computers11030041.

[28] O. Navarro Ferrer, "Analysis of reinforcement learning techniques applied to honeypot systems," 2021, [Online]. Available: http://hdl.handle.net/10609/126948

[29] J. S. López-Yépez and A. Fagette, "Increasing attacker engagement on SSH honeypots using semantic embeddings of cyber-attack patterns and deep reinforcement learning," *Proc. 2022 IEEE Symp. Ser. Comput. Intell. SSCI 2022*, pp. 389–395, 2022, doi: 10.1109/SSCI51031.2022.10022206.

[30] H. Van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-learning".

[31] M. A. Kristyanto, H. Studiawan, and B. A. Pratomo, "Evaluation of Reinforcement Learning Algorithm on SSH Honeypot," *Proceeding - 6th Int. Conf. Inf. Technol. Inf. Syst. Electr. Eng. Appl. Data Sci. Artif. Intell. Technol. Environ. Sustain. ICITISEE 2022*, pp. 346–350, 2022, doi: 10.1109/ICITISEE57756.2022.10057816.

[32] A. Pashaei, M. E. Akbari, M. Zolfy Lighvan, and A. Charmin, "Early Intrusion Detection System using honeypot for industrial control networks," *Results Eng.*, vol. 16, no. July, p. 100576, 2022, doi: 10.1016/j.rineng.2022.100576.