mathematics

MDPI

Article

AARF: Autonomous Attack Response Framework for Honeypots to Enhance Interaction Based on Multi-Agent Dynamic Game

Le Wang [1,2], Jianyu Deng [1], Haonan Tan [1], Yinghui Xu [1], Junyi Zhu [1], Zhiqiang Zhang [3], Zhaohua Li [4], Rufeng Zhan [1] and Zhaoquan Gu [2,3,*]

[1] Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou 510006, China; wangle@gzhu.edu.cn (L.W.); dengjianyu@e.gzhu.edu.cn (J.D.); 2112106206@e.gzhu.edu.cn (H.T.); 2112233150@e.gzhu.edu.cn (Y.X.); 2112233067@e.gzhu.edu.cn (J.Z.); 2006400027@e.gzhu.edu.cn (R.Z.)
[2] Department of New Networks, Peng Cheng Laboratory, Shenzhen 518055, China
[3] School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), Shenzhen 518055, China; 23b951049@stu.hit.edu.cn
[4] Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen 518000, China; 202312281023@std.uestc.edu.cn
* Correspondence: guzhaoquan@hit.edu.cn

Abstract: Highly interactive honeypots can form reliable connections by responding to attackers to delay and capture intranet attacks. However, current research focuses on modeling the attacker as part of the environment and defining single-step attack actions by simulation to study the interaction of honeypots. It ignores the iterative nature of the attack and defense game, which is inconsistent with the correlative and sequential nature of actions in real attacks. These limitations lead to insufficient interaction of the honeypot response strategies generated by the study, making it difficult to support effective and continuous games with attack behaviors. In this paper, we propose an autonomous attack response framework (named AARF) to enhance interaction based on multi-agent dynamic games. AARF consists of three parts: a virtual honeynet environment, attack agents, and defense agents. Attack agents are modeled to generate multi-step attack chains based on a Hidden Markov Model (HMM) combined with the generic threat framework ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge). The defense agents iteratively interact with the attack behavior chain based on reinforcement learning (RL) to learn to generate honeypot optimal response strategies. Aiming at the sample utilization inefficiency problem of random uniform sampling widely used in RL, we propose the dynamic value label sampling (DVLS) method in the dynamic environment. DVLS can effectively improve the sample utilization during the experience replay phase and thus improve the learning efficiency of honeypot agents under the RL framework. We further couple it with a classic DQN to replace the traditional random uniform sampling method. Based on AARF, we instantiate different functional honeypot models for deception in intranet scenarios. In the simulation environment, honeypots collaboratively respond to multi-step intranet attack chains to defend against these attacks, which demonstrates the effectiveness of AARF. The average cumulative reward of the DQN with DVLS is beyond eight percent, and the convergence speed is improved by five percent compared to a classic DQN.

Keywords: honeypot; interaction; multi-agent; attack chain; value label sampling; reinforcement learning

MSC: 37M05

Citation: Wang, L.; Deng, J.; Tan, H.; Xu, Y.; Zhu, J.; Zhang, Z.; Li, Z.; Zhan, R.; Gu, Z. AARF: Autonomous Attack Response Framework for Honeypots to Enhance Interaction Based on Multi-Agent Dynamic Game. *Mathematics* **2024**, *12*, 1508. https://doi.org/10.3390/math12101508

Academic Editor: Cheng-Chi Lee

Received: 22 March 2024
Revised: 8 May 2024
Accepted: 8 May 2024
Published: 11 May 2024

1. Introduction

Traditional defense strategies include firewalls [1], intrusion detection [2], anti-virus [3], etc. Their main purpose is to alert and prevent network attacks, but they lack initiative. As a deception defense strategy, honeypots can actively attract attackers and collect and

analyze specific attack behaviors. Honeypots interact with attackers attempting to penetrate the corporate intranet for lateral movement to delay and capture attacks. Honeypots and attackers with different goals have a competitive relationship. It can be seen that the interaction between the honeypot and attacker is actually a game process, and the high interaction with the honeypot is built on each effective response action. The more the honeypot interacts with the attacker, and the more the pattern of interaction matches the attack behavior, the more effective the deception defense. Honeypots with a high interaction strategy are valuable for deception defense in network security.

There is much research on honeypots focused on emulation improvement. For example, this is often accomplished by enhancing the authenticity of honeypot operating systems, applications, and their vulnerabilities [4–6] to match existing business systems. However, simply enhancing the simulation of honeypots is limited when considering the persistence of attack interactions, which lacks any modeling of attack response actions. More attention is useful in the generation of rational response strategies for honeypots. During attack and defense games, a honeypot is able to generate and adjust its response strategy through learning, which directly affects the maintenance of the interaction relationship. Thus, modeling attackers is an effective way to enhance the honeypot interaction. Recent research has typically modeled the attacker as part of the environment and defined single-step attack actions by simulation, which can simplify the complexity of real attacks and does not adequately express the variety of attack strategies and behaviors. This makes it difficult to realistically describe the mechanisms associated with an attack. For instance, in [7], attackers are modeled into custom network environments, and the generated attack strategies are not sufficiently reflective of real attack characteristics and are difficult to apply to other network scenarios. Meanwhile, most of the current attack action modeling used to train honeypot agents is mostly independent and cannot constitute a complete attack chain. In [8], the authors focus on honeypot techniques for SQL injection and XSS attacks and do not consider response strategies under the correlation of actions in a complete attack chain. These limitations result in an insufficient interaction of the response strategy generated by the honeypot, making it difficult to establish a continuously interactive connection with attackers in a dynamic environment.

To tackle these challenges, we propose an autonomous attack response framework, AARF, based on a multi-agent dynamic game to model each agent. The AARF framework consists of three parts, where the environment part is provided by a customized virtual honeynet, HoneypotSim, the attack agent can automatically generate a valid attack chain based on an HMM, and the honeypot agent is constructed based on RL with a response strategy. The core idea is to model the dynamic attack and defense game process based on RL, through the dynamic interaction between the honeypot agents and the environment to generate an interactive response strategy when facing a multi-step attack chain. Based on the results of attack and defense interactions, honeypot response strategies and attack chain generation strategies are automatically updated. We divide the whole process of deception defense into three phases and modeled three types of honeypots responsible for different phases of targeting collaborative response attacks. The attack agents aim at generating effective attack chains, and the process is divided into three stages: seed attack chain generation, dynamic attack chain evolution, and valid attack chain validation. The honeypot agents accomplish the task of interacting with the attacker by sensing state changes and choosing the optimal response action. Considering the highly dynamic nature of real network environments, we introduce a dynamic reward mechanism and a dynamic honeypot anti-identification capability. In the training of RL agents, the experience replay technique based on random uniform sampling tends to ignore high-value sample experiences in a dynamic multi-honeypot environment. In order to address the data sampling problem and to improve the efficiency of learning by agents, we propose the DVLS algorithm. The goal is achieved by calculating the probability value of information provided to the memory experiences of the learning process. We further couple the DVLS with the DQN method to replace the traditional random uniform sampling method and

improve the overall training effect of the DQN algorithm. We test AARF and a DQN with DVLS on our customized environment where the multi-honeypot is gamed with an attacker. AARF is able to effectively interact with attack actions and trap the complete attack chain with a reasonable response strategy. The DQN with DVLS consistently outperforms baseline RL algorithms. Based on the above analysis, the contributions of our paper can be summarized as follows:

- We develop an RL-based framework, named AARF, which successfully correlates honeynet environments, attack agents, and honeypot agents based on dynamic game modeling. AARF can autonomously make response decisions to interact with the attack chain to achieve deception defense in a multi-agent environment.
- We model the attack agent based on an HMM by combining the ATT&CK techniques. Attack agent rapidly generates valid attack chains according to seed chain generation, dynamic chain evolution, and valid chain verification in three stages, which support training honeypot agents.
- We design a customized multi-honeypot environment for RL training, HoneypotSim, including three functional types of honeypots.
- We propose a dynamic value label sampling (DVLS) algorithm that probabilizes the importance of experience samples to improve sample utilization, and we successfully replace the sampling method of the DQN to improve training efficiency.

The remainder of this paper is organized as follows. In Section 2, we give a brief overview of the related research on modeling attack agents and honeypot agents. Section 3 illustrates the relevant preliminaries of our framework. Section 4 introduces the modeling process using AARF and its various modules. Section 5 presents the experimental configuration and results. Finally, the conclusion and future work are summarized in Section 6.

2. Related Work

In the dynamic game of cybersecurity, attack agents and defense agents are two core elements. In recent years, more and more research works have modeled automated attackers based on machine learning methods. For defense agents, honeypot technology, which is initiative and capable of achieving deceptive defense goals, has attracted more researchers' attention.

2.1. Attack Agent Modeling

In terms of SQL injection attacks, refs. [9,10] explore how to adopt optimal strategies based on environment changes to efficiently exploit vulnerabilities based on RL by simulating this specific scenario. They modeled the attack behavior as a Markov decision process with the task goal of executing an effective SQL injection action. Concerning research on modeling XSS attacks, ref. [11] model XSS payload generation as a hierarchical RL problem, so that agents can learn to fuzz new source–sink combinations, generating different payloads and bypassing the associated sanitization. The above studies are all aimed at modeling single-step attacks in specific attack scenarios and cannot provide a complete attack chain strategy with temporal relationships.

CybORG [12] is a work-in-progress gym for autonomous cyberoperations. They modeled red team agents based on a DDQN to complete penetration testing in simulated CTF scenarios that feature a network of vulnerable hosts connected to different subnets. CyberBattleSim [7], an AI attack and defense simulation tool open-sourced by Microsoft in 2021, utilizes a reinforcement learning approach for automated intranet penetration and is used to study the interaction process of automated agents in a simulated abstract enterprise network environment. Ref. [13] proposes a generalized solution for modeling framework-based attacks based on the blackboard architecture. It combines rules and facts that implement attack type determination and attack decisions that go beyond the deployment of a single exploit against a single identified target. The attack agents or threat models constructed in the above studies do not take into account the data sources in actual

attack scenarios. Therefore, the generated attack strategy lacks authenticity and is difficult to use in actual attack applications.

2.2. Honeypot Agent Modeling

HoneyMustard [4] is an application-level real-time user behavior simulation framework for improving the fidelity of honeypot systems. It collects logical user action sequences of various applications to build datasets and utilizes computer vision techniques to simulate user activities. Ref. [5] proposed HoneyPLC, a highly interactive, scalable, and malware-collecting honeypot in a Programmable Logic Controllers (PLCs) scenario. It is based on honeyd, the snap7 framework combined with nmap, snmpsim, and other programs to implement simulation modules for different protocols, and all code injected by the attacker is captured within the repository module. FirmPot [6] is a framework for generating intelligent interactive honeypots by simulating different firmware, which can respond to firmware requests from attackers. Refs. [4–6] all focus on improving the function to get a high-fidelity honeypot. By making the simulated user activity interface more realistic, ref. [4] attracts attacks from attackers but does not establish any actual interaction with the attacker. In [5,6], the authors improve the honeypot's interactions by simulating the response of a real system. However, they are all simulations under specific protocols and are static responses.

NeuralPot [14] leveraged pcap files to train a Deep Neural Network (DNN) to generate modbus network traffic and combined the DNN with the databus system of Conpot to generate more realistic response traffic in order to improve the interactions of honeypot. Ref. [15] trained a bidirectional encoder representation from transformers (BERT) using HTTP protocol datasets. The results of the BERT model output were used as action candidates in the Markov decision process (MDP), so that the honeypot could effectively respond to the client based on received requests, rather than responding to fixed requests. Based on the semi-Markov decision process (SMDP) method, ref. [16] modeled the random transfer and sojourn time of the attacker in a honeynet to generate an adaptive response strategy, improving the interaction time with the attacker. Ref. [17], based on [18], proposed an SSH self-adaptive honeypot, which modeled the honeypot's response actions and used a Deep Q-Network algorithm training to determine how to interact with external attackers. Ref. [19] followed the same model and algorithm as [17] and evaluated the interaction duration. It was concluded that adaptive honeypots could obtain more information than ordinary honeypots, resulting in the interaction time between the attacker and the honeypot being longer. This provides strong experimental support for our use of reinforcement learning to improve the interaction time between the honeypot and the attacker. In [14–17], the authors employ advanced techniques such as DNNs and RL to address the limited interactions in traditional honeypots. Refs. [14,15,17] are more focused on the honeypot's own response research, without additional analysis of the attacker's attack information. This results in an inability to capture the attacker's behavioral patterns. In [16], the transfer probability derived from theory may be different from the real data distribution, so it may not obtain a better corresponding strategy.

In this paper, we model the attacker and the defender separately in a multiple-honeypot scenario. For attackers, we complete the generation of the valid attack chain based on an HMM using the real attack probability as input. This solves the problem of attackers being modeled into the environment and simulating single-step attacks. For defenders, we focus on the optimization problem of honeypot response strategies in dynamic game scenarios based on RL and finally generate response strategies for adaptive scenarios. This solves the problem of the static simulation and response of honeypots in the above studies. Our work overall enhances the effective interaction of honeypots against attacks.

3. Preliminaries

In this section, we introduce the background knowledge and underlying concepts of the techniques used in this paper.

3.1. Lateral Movement and ATT&CK

Lateral movement often occurs in the later stages of network penetration in an enterprise intranet environment. It refers to a series of threatening measures taken by an attacker in search of higher-value assets after the attacker has already gained initial access, and common actions include implanting malware, elevating privileges, and so on. Lateral movement has the characteristics of a large threat surface and strong threat. Therefore, attack detection against intranets [20–22] is a key research priority in the field of network security.

MITRE ATT&CK [23] is a technology framework encompassing over 200 techniques that attackers may employ during an attack based on real-world observation. The ATT&CK model describes more comprehensively the various attack behaviors adopted by the attacker during the whole process of network attack. Thus, it is useful for threat intelligence analysis, security tool evaluation, and defense strategy development. The framework provides a common standardized language based on tactics, techniques, and procedures (TTP), where techniques denote specific actions of an attacker to achieve a tactical goal. Therefore, in this paper, we think of an ordered logical combination of multi-step attack techniques as an attack chain. Similarly, the combination of attack techniques in the lateral movement phase of the intranet is the intranet attack chain. The attacker realizes the attack goal by executing the multi-step attack chain.

3.2. Honeypot and Honeynet

A honeypot is a proactive, aggressive network spoofing defense technology. It is able to simulate real services to attract attackers to attack and collect attack data, analyze attack behavior, and protect real systems from attacks [24–26]. A network environment where multiple honeypots are intricately deployed is known as a honeynet. As shown in Figure 1, when the attacker accesses a normal intranet environment, malicious attacks are introduced into the honeynet environment through techniques such as IP redirection. Honeypots guide attackers to roam within the honeynet by responding reasonably to attack requests. This enables effective capture of attack information and defense operations such as tracking.

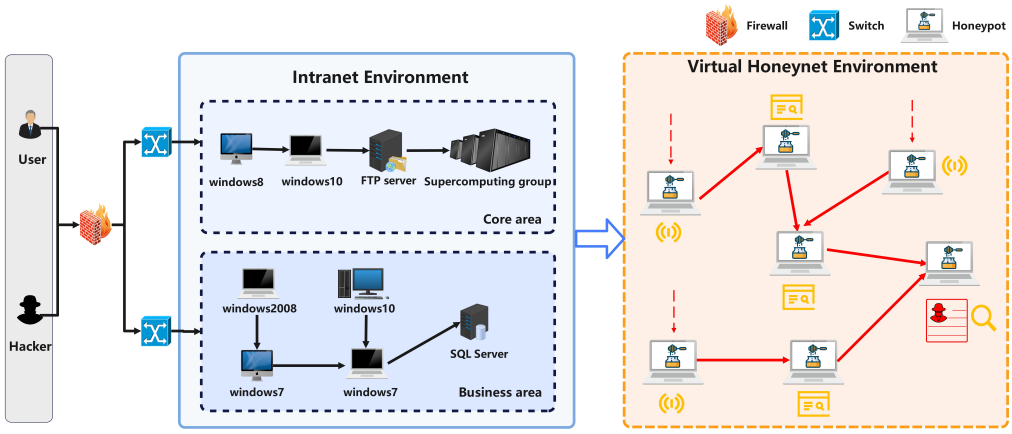


Figure 1. Application scenarios: Honeynet mapping for intranet environment. After the attackers enter the intranet scenario, they are guided into the honeynet environment. The honeynet is a service mapping of the actual intranet, in which a large number of honeypots containing false information are deployed.

3.3. Hidden Markov Model

A Hidden Markov Model (HMM) is a probabilistic model of temporal sequences to describe sequences of observations randomly generated by hidden Markov chains. In the HMM, the state of the system is not visible, but the state-generated observations are visible. HMM is determined by three main elements, the initial state probability vector π, the state transfer probability matrix A, and the observation probability matrix B. Typically, a Hidden Markov Model can be succinctly represented by the triad $\lambda = (A, B, \pi)$. A is the state

transfer probability matrix $[a_{ij}]_{N\times N}$, where a_{ij} denotes the probability of transferring from the condition of being in the state q_i at moment t to the state q_j at moment $t+1$, which is the transition probability.

$$a_{ij} = P(i_{t+1} = q_j | i_t = q_i), i = 1, 2, \ldots, N; j = 1, 2, \ldots, N \quad (1)$$

B is the observation probability matrix $B = [b_j(k)]_{N\times M}$, where $b_j(k)$ denotes the probability of generating an observation v_k under the condition of being in state q_j at moment t, which is the emission probability.

$$b_j(k) = P(o_t = v_k | i_t = q_j), k = 1, 2, \ldots, M; j = 1, 2, \ldots, N \quad (2)$$

π is the initial state probability vector, where π_i is the probability of being in state q_i at moment $t = 1$, which is the initial state probability.

$$\pi_i = P(i_1 = q_i), i = 1, 2, \ldots, N \quad (3)$$

4. Method

Our work focuses on the lateral movement stage within an enterprise intranet by simulating the deployment of honeypots within a virtual honeynet environment. As shown in Figure 1, attackers accessing the intranet services from the external Internet are directed to our customized honeynet environment HoneypotSim. In HoneypotSim, numerous deceptive false targets and misleading information are deployed. Attackers may deviate from their attack path and be guided to access pre-configured honeypot nodes one by one.

AARF is a robust, highly interactive collaborative response framework for a dynamic attack–defense game that fights in a multi-honeypot scenario. There is an adversarial relationship between the attack and defense agents and a functional collaborative relationship between the honeypots. The honeypotSim module, defense agents module, and attack agents module are the main components of AARF. The specifics of this AARF framework are detailed for each module in Figure 2.

In our custom-designed multi-honeypot environment, HoneypotSim supports gaming by simulating protocols, ports, and other services. For example, the SSH protocol service corresponds to port 22, the Telnet protocol to port 23, and so on. To enhance the realism of the attacker's modeling, threat intelligence and attack logs are used as attack data input to calculate the frequency of attack techniques. ATT&CK provides attack action support for attack agent modeling. The attack agent orchestrates and combines the tactical attack techniques in ATT&CK based on the HMM method to generate valid attack chains. The objective of attack agents is to execute each step of the attack actions smoothly according to the generated reliable attack chain, aiming to gain more permissions on hosts. The honeypots, acting as defense agents guided by the RL control engine, engage in a game with attack agents. Each honeypot agent is capable of awareness and anti-identification. Based on the establishment of interactive responses, honeypot agents can capture more information about the attackers, achieving effective deception and isolation. Figure 3 demonstrates the specific process of modeling attack–defense interactions based on RL and the generation of honeypot optimal response strategies. RL is an automatic learning framework that follows the behavior, feedback, and motivation model. It can effectively model automated decision-making problems in dynamic network environments by constructing agents that iteratively interact with the environment and continuously optimize their behavioral strategies. In the AARF framework, the honeypot faces a multi-step attack behavior chain based on RL to explore more response actions in the action space in the pre-training period. Meanwhile, it iteratively optimizes the response action strategy based on state observations and reward feedback from the environment. In the late stage of training, the honeypot agents continuously adapt to the dynamic honeynet environment and multi-step attack chain changes, more rationally utilize the response actions, and ultimately generate the optimal honeypot response strategy. Based on the reasonable response to

attract attack agents, the framework thus improves the efficiency of a successful response and enhances the interaction between the honeypot and the attack agents.

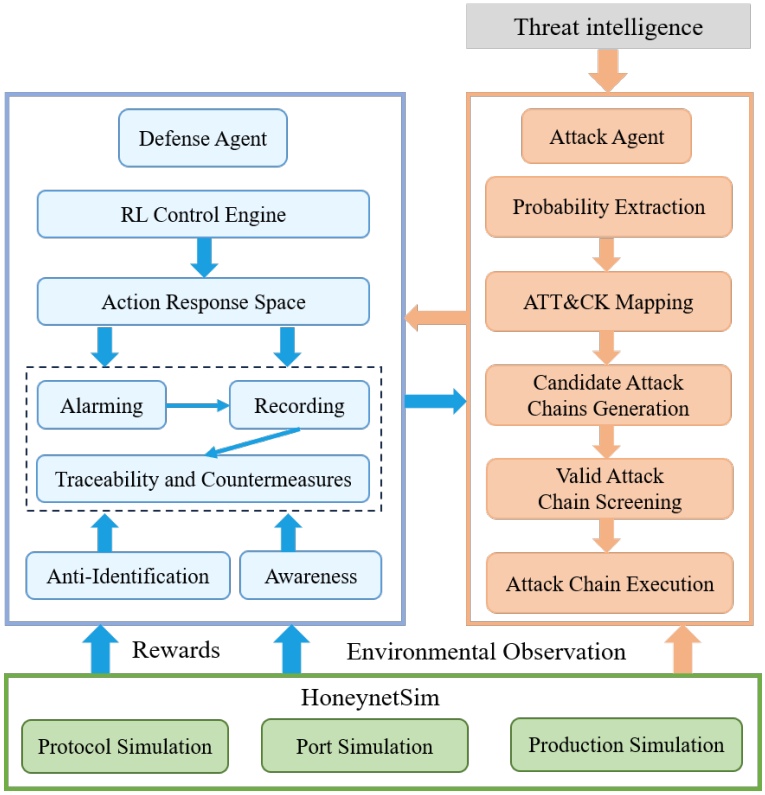


Figure 2. Detailed framework of AARF. With the support of the virtual honeynet environment HoneypotSim, the attack agent and the defense agent complete the game interaction and automatically generate the optimal attack response strategy for the honeypot.

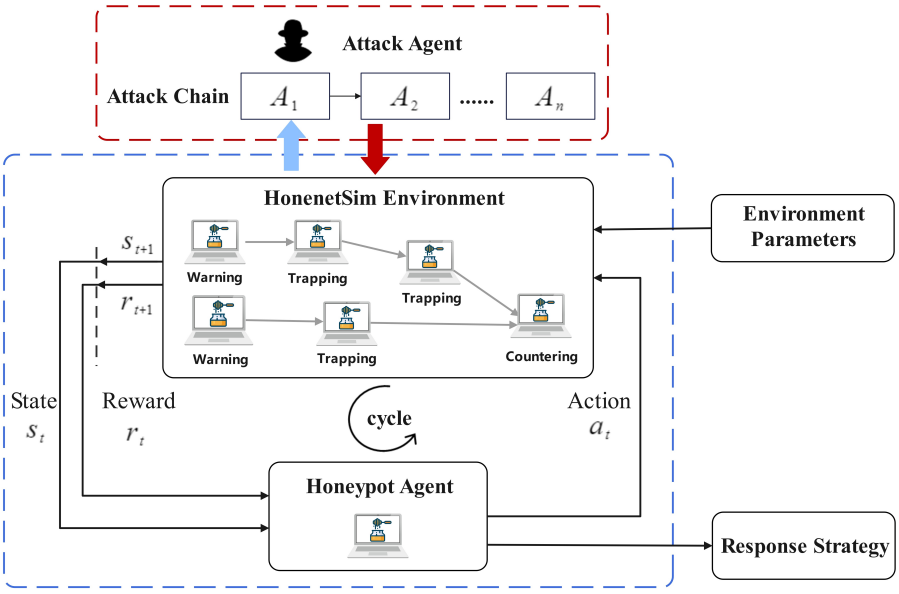


Figure 3. Attack and defense interaction modeling. Honeypot agents interact with multi-step attack chains in the honeynet environment. The honeypot agents iteratively generate optimal response strategies based on state observations and feedback rewards. The red and blue arrows indicate attack and defense actions respectively.

4.1. Honeypot Models

The general deception defense process can be divided into three phases: threat warning, information collection, and traceability countermeasures. In this paper, we design three different types of honeypots according to these three phases, respectively, each serving a distinct purpose, yet working together in a coordinated manner to achieve deception defense. Through the collaborative response of functional honeypots, we ensure the maximum utilization of each honeypot's capabilities, resulting in more efficient defense.

Warning honeypots are deployed at the network edge, when attacked, they immediately communicate to neighboring nodes to alert attack-related information. Trapping honeypots are deployed inside the network; they record the attack behavior to help analyze the purpose of the attacks. Countering honeypots are deployed at the core of the network; after obtaining sufficient attack information, they complete source traceability countermeasures. The deployment of each honeypot is shown in Figure 4. Each honeypot has the same response actions, with different functional actions reflecting their characteristics. Based on the different deployment locations, they each respond to different stages of the attack chain. All the defensive functional actions are completed based on the response success.

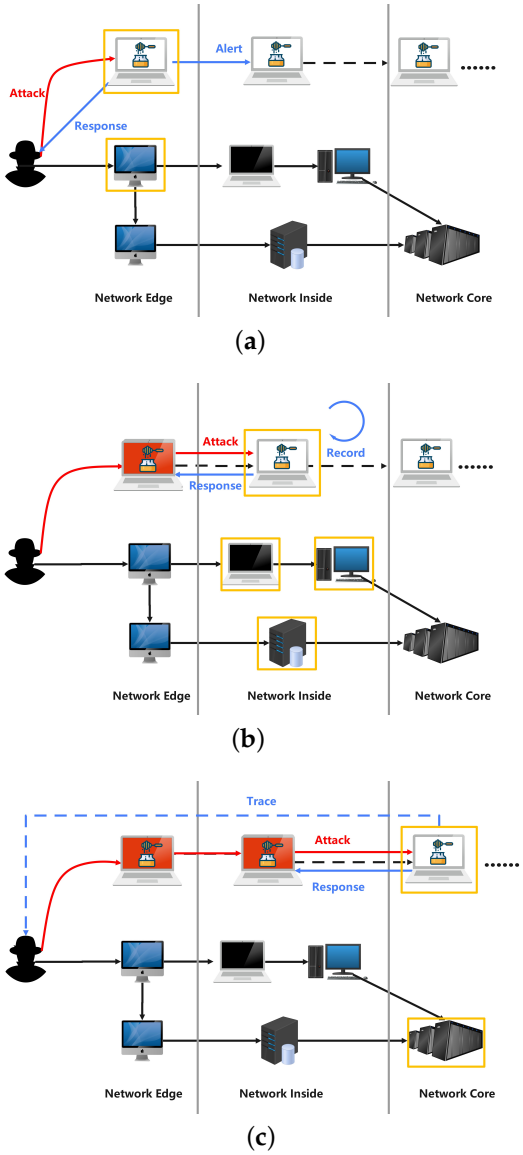


Figure 4. Honeypot classification: (**a**) warning honeypot deployment; (**b**) trapping honeypot deployment; (**c**) countering honeypot deployment.

4.2. Automatic Attack Chain Generation

Attack agents can automatically and quickly generate valid attack chains. This process can be divided into three main phases, seed attack chain generation, dynamic attack chain evolution, and valid attack chain verification. In this paper, we generated a seed attack chain based on an HMM accomplished by orchestrating attack techniques, dynamically evolved the attack chain based on the seed chain, and stored it in the candidate attack chain collection. Finally, valid attack chains were quickly verified under the honeynet scenario interaction. The specific steps for generating valid attack chains are illustrated in Figure 5.

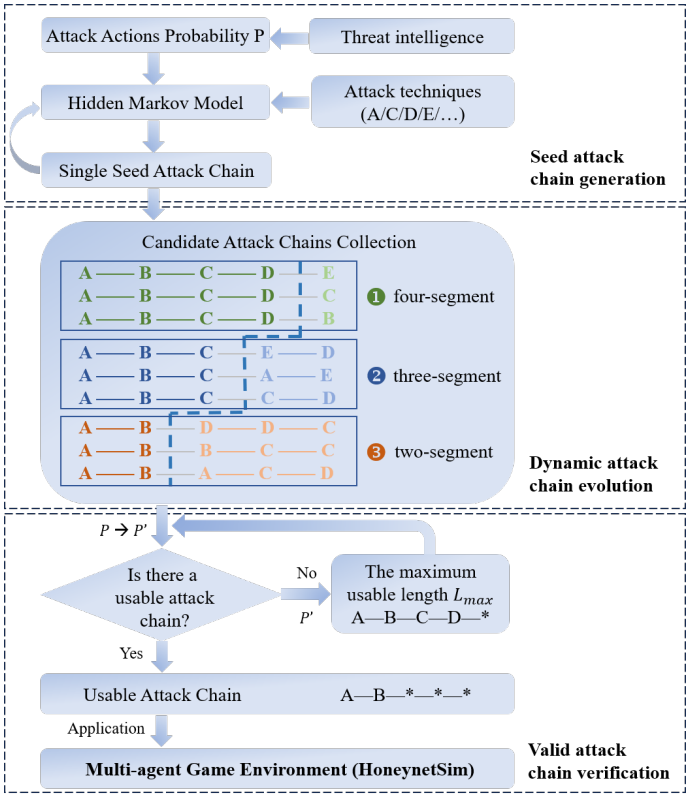


Figure 5. Valid attack chain generation. The first step is to generate a seed attack chain based on an HMM. The second step is to evolve the attack chains based on the seed chain and store them in the candidate attack chain collection. The third step is to screen valid attack chains in the simulated network environment. The capital letters A to E in the figure represent the attack techniques in ATT&CK, and they are combined to form an attack chain. The * in the attack chain indicates the pending attack action, which will be filled according to the actual situation.

4.2.1. Seed Attack Chain Generation

An HMM is a sequence model that effectively captures dependencies and sequential relationships between events. The natural inclusion of tactics and technology in the ATT&CK matrix makes it possible to rationally represent transitions between technology and tactics using an HMM. In this case, techniques can be viewed as the observed state and tactics as the hidden state, where a series of techniques are implemented to achieve tactical goals. To generate valid attack chains, we arranged ATT&CK techniques to describe attack events logically based on a first-order HMM probabilistic model, as shown in Figure 6. The following probabilities were used as the input to the HMM.

- Initial state probability (π_i): the probability of starting with each tactic.
- Transition probability (a_{ij}): the probability of transitioning between each pair of tactics.
- Emission probability ($b_j(k)$): the probability of each tactic containing specific techniques.

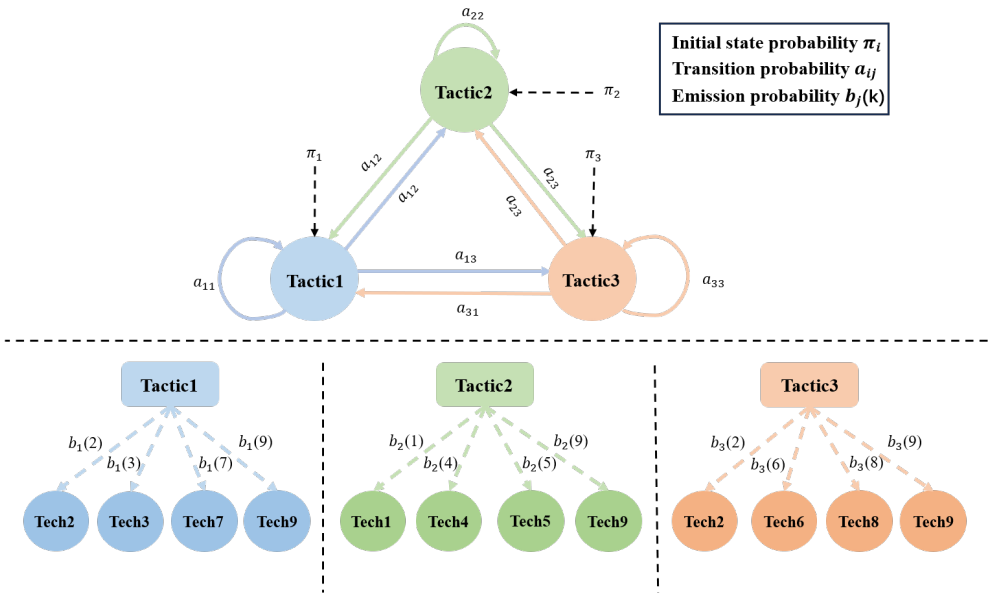


Figure 6. The first order HMM. Elements in the model include ATT&CK tactics and techniques. The attack probability set includes the initial state probability of tactics, the transition probability between tactics, and the emission probability of techniques included in the tactics.

To align with the model construction, nine techniques in the ATT&CK lateral movement stage were classified into three tactics based on their performance in real threat intelligence as shown in the Table 1. Based on statistical methods for threat intelligence, we can calculate the frequency of transition between the three tactics and the frequency of selection of each technique to use. The first attack chain generated based on the input attack probability set is the one with the highest success rate considered by the model at that time and is called a seed attack chain. By adjusting the three probabilities of the HMM parameters, we can also automate the generation of attack chains that reflected different attack goals. (This paper refers to some open source threat intelligence: https://github.com/mitre-attack/attack-stix-data, accessed on 21 March 2024).

Table 1. Attack techniques' classification.

Classification	ID	Describe
Tactics1	Technique2-T1534	Internal spear phishing
	Technique3-T1570	Lateral tool transfer
	Technique7-T1072	Software deployment tools
	Technique9-T1550	Use alternate authentication material
Tactics2	Technique1-T1210	Exploitation of remote services
	Technique4-T1563	Remote service session hijacking
	Technique5-T1021	Remote services
	Technique9-T1550	Use alternate authentication material
Tactics3	Technique2-T1534	Internal spear phishing
	Technique6-T1091	Replication through removable media
	Technique8-T1080	Taint shared content
	Technique9-T1550	Use alternate authentication material

4.2.2. Dynamic Attack Chain Evolution

A single attack chain generated based on the HMM is not necessarily available, and this paper argues that it is a time-consuming process to call the HMM again to regenerate the attack chain if the current one is not available. Therefore, in order to save time and improve the efficiency of screening valid attack chains, we constructed a collection of candidate attack chains based on the dynamic evolution of the seed attack chain.

As shown in the Figure 6, the collection of candidate attack chains consists of three types: four-segment matching, three-segment matching, and two-segment matching, which

means the same length of attack chains from left to right. The core idea of the evolution process is to reconstruct some attack actions based on the seed attack chain and store them in the candidate attack chain collection. This provides the attacking agent with a diverse search space to improve the success rate of valid attack chains. In this process, it is not necessary to regenerate the multi-step attack chain based on the HMM for storage each time. Instead, certain attack steps are adjusted according to the first stored attack chain (seed attack chain), thus ensuring the success rate and availability of a valid attack chain.

4.2.3. Valid Attack Chain Verification

The verification of the valid attack chain is performed based on the interaction with host nodes configured with vulnerability information in a honeynet environment. The attack chains are chosen according to their storage order based on the collection of candidate attack chains. When an attack step is unusable, its probability is reduced by a fixed proportion and allocated to other probabilities. The probability set P is continually adjusted to $P'\left\{p, t, \alpha'\right\}$ based on attack execution feedback. During the verification process, the maximum length L_i available for each candidate attack chain is recorded. If a complete and valid attack chain exists, it is chosen to participate in the training of the attack–defense game. On the contrary, there may be no valid attack chain in the set of candidate attack sequences. In this case, we select the attack chain with the maximum usable length L_{max} among them as the target attack chain and adjust it locally according to the new probability. This process continues until a valid attack chain is obtained to complete the attack–defense interaction.

Our proposed valid attack chain generation method can obtain usable attack chains more efficiently and quickly. This can increase the efficiency of automated attacks, thus accelerating the training process of honeypot agents in attack and defense games.

4.3. Dynamic Attack–Defense Game in a Multi-Honeypot Environment

4.3.1. Dynamic Honeypot Agent

The attack agent executes an attack action based on its own partially observed state, triggers a honeypot response, and decides on the execution of subsequent actions according to the response result. The honeypot agents respond to the attack action and adjust their response strategy based on the next observed state and reward values. The cyclic interaction between the attack agent and the honeypot is modeled as a multi-agent Markov game process based on AARF. Finally, the attack agent and the defense agent obtain their respective optimization strategies.

In real production scenarios, the value of assets is gradually increasing on the path from the network edge to the network core. Therefore, the attacker intrudes further, it gets closer to sensitive data, and a defense response matters more. To obtain a more effective strategy output, in the game modeling, we adjust the reward distribution based on the dynamic reward mechanism. The reward values are set not only based on the cost of configuring different honeypot types C_{type} but are also related to the progress of the attacker's intrusion T. This encourages the defender to make more rational response actions. Equation (4) illustrates the formation of honeypot's reward R_t, where T represents the process of attack, N_a represents the number of attacks, μ represents the reward factor, and ω is a constant. (The * in all equations in this paper represents the multiplication sign).

$$R_t = C_{type} + \mu * (T + N_a) + \omega \tag{4}$$

At the same time, attackers have the opportunity to identify honeypot nodes during the attack attempt, and honeypots also use anti-identification configurations to better hide themselves. We quantify the anti-identification configuration of the honeypot into a probabilistic representation of the anti-identification capability. The worse the honeypot's response action is, the higher the probability of an attacker seeing through it, which means the lower the honeypot's anti-identification ability. Similarly, the more reasonable the

response action of the honeypot agent is, the higher the feedback reward value it receives, and the higher the anti-identification ability is. $V(e_i) = \max(\min(r_t, c), -c)$ represents the reward value interval mapping for the range $[-c, c]$, where e_i represents the i-th experience. Then, based on the *sigmoid* function, the anti-identification probability is normalized to be in the range (a, b), $(0 < a < b < 1)$.

$$P_h^t = (b - a) * sigmoid\left(\frac{r_t - \min(V(e_i))}{\max(V(e_i)) - \min(V(e_i))}\right) + a \tag{5}$$

4.3.2. Dynamic Value Label Sampling in a Multi-Honeypot Environment

The honeynet environment is characterized by non-static features such as dynamic reward mechanisms, dynamic anti-identification configurations, etc. The honeypot agent stored training data based on RL during the training process are large and varied. The experience replay mechanism in RL can mix and store different experiences during the training process. The agent remembers and reuses past experience samples, breaking the correlation of continuous sample sequences, and making the learning process more stable and efficient. However, the general uniform sampling method faces the problem of being unable to instantly sample important information for learning. This method randomly samples a certain batch of experience data based on the experience samples stored in the experience buffer during each sampling process to learn. This neglects the learning of high-value samples, which reduces the sampling efficiency and learning efficiency of the agent. Therefore, we propose a dynamic value sampling method in a multi-honeypot environment. In each sampling process, the agent can sample samples with higher importance and value for learning, so it can effectively improve the sampling efficiency and accelerate the convergence of RL.

The anti-identification ability is calculated based on the feedback reward. The larger its value, the more effective the interaction at that time, and the higher the value of the experience sample at that moment. Therefore, we characterize the relative size of the value of stored experience samples based on the anti-identification probability value P_h^t at time t of the honeypot. Based on this, each piece of experience data is marked with a value label.

$$L_k = P_h^t + \varepsilon \tag{6}$$

Here, ε is a small constant used to prevent samples from being neglected at the edge case where the reward value is zero, ensuring that all samples are labeled.

However, if we just sample the data according to the value label, it leads to some sample data with low label values to never be selected. The lack of diversity in experiences leads to an insufficient generalization of learning strategies. To alleviate this problem, we use probabilities to represent the values of different sample data. It ensures that the empirical data of the smallest label are also non-zero and has a chance to be selected, effectively ensuring sampling diversity. Specifically, we define the sampling probability as follows:

$$P_i = \frac{L_i^k}{\sum_0^m L_j} \tag{7}$$

When performing value-based data sampling, different samples are assigned different selection probabilities based on the honeypot's anti-identification capabilities. The original expected distribution is also changed, and the final expected value estimate of the random sample becomes a biased estimate. The resulting bias problem can be mitigated by using importance sampling (IS) weights.

$$\omega_j = (\frac{1}{(N \cdot P(j))})^\beta \tag{8}$$

The details of the value sampling method based on dynamic rewards and anti-identification capabilities in multiple honeypot scenarios are shown in Algorithm 1.

Algorithm 1 Dynamic value label sampling (DVLS)

Input: experience replay memory D, i-th experience (s, a, r_t, s'), L_j, batch, expoments β, constants a, b, c, ε. Initialize $h \in \{1, 2, 3\}$

Update the sampling probability of experience $P_{i-1} = \frac{L_{i-1}^k}{\Sigma_0^m L_j}$

Sample experience transition $B(\varphi_j, a_j, r_j, \varphi_{j+1}) \overset{P_i}{\leftarrow}$ SampleBatch $(D, batch)$

Compute IS weight $\omega_j = (\frac{1}{(N \cdot P(j))})^{\beta}$

Zoom experience information reward value $V(e_i) = \max(\min(r_t, c), -c)$

Calculate anti-identification probability:

$$P_h^t = (b - a) * \mathrm{s}\,igmoid\left(\frac{r_t - \min(V(e_i))}{\max(V(e_i)) - \min(V(e_i))}\right) + a$$

Label experience transition L_i based on $P_h^t \quad L_i = P_h^t + \varepsilon$

4.3.3. DQN with Dynamic Value Label Sampling

Deep Q-Network (DQN) [27] is an RL algorithm based on deep learning (DL). The classic DQN algorithm uses random uniform sampling, but traditional random sampling methods can lead to a failure to consider the potential value of experiences and ignore the learning of important experiences. Especially in attack and defense scenarios, it is difficult to accurately learn high-value experience information in the face of high-dimensional and complex attack state features. As a result, it reduces the effective utilization of samples, which can impact the training quality and convergence speed of the model.

Due to the problems of random uniform sampling, it may require a larger number of samples to achieve the same effect as importance sampling. This can enhance training time and computational costs. We coupled the dynamic value label sampling method with the DQN method to replace the traditional random uniform sampling method and improve the overall training effect of the DQN algorithm. Targeted sampling was performed according to the value of experience during the training process, and the value labels were updated regularly during training. Utilizing this sampling method can improve the effective utilization of samples. This is particularly useful in some RL problems where the estimation of certain state–action pairs may be more accurate and can provide better prior information. Algorithm 2 provides a detailed implementation of DQN with the DVLS algorithm.

Algorithm 2 DQN with dynamic value label sampling

Input: L_j, exponents β, constant a, b, c, ε ; Initialize replay memory D to capacity $N, h = 1$; Initialize action-value function Q with random weights

for $episode = 1 \rightarrow M$ **do**
  Initialize sequence $s_1 = \{x_1\}$
  for $t = 1 \rightarrow T$ **do**
    With probability select a random action a_t
    otherwise select $a_t = \max_a Q^*(\varphi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\varphi_{t+1} = \varphi(s_{t+1})$
    Store transition i $(\varphi_t, a_t, r_t, \varphi_{t+1})$ in D
    $P_i, \omega_j, V(e_i) = \mathrm{DVLS}(i, L_{i-1}, \beta, a, b, c)$
    Compute TD-error $\delta_j = r_j + \gamma \max_{a'} Q(s_j, a_j) - Q(s_{j-1}, a_{j-1})$
    Set $y_j = \begin{cases} r_j & for\ terminal\ \varphi_{j+1} \\ r_j + \gamma \max_{a'} Q(\varphi_{j+1}, a'; \theta) & for\ \text{non-}terminal\ \varphi_{j+1} \end{cases}$
    Perform a gradient descent step on $P_i, \omega_j, V(e_i) = \mathrm{DVLS}(i, L_{i-1}, \beta, a, b, c)$
    Accumulate weight change $\Delta \leftarrow \Delta + \omega_j \cdot \delta_j \cdot \nabla_\theta Q(s_{j-1}, a_{j-1})$
  end for
end for

5. Experiment

The abstract HoneypotSim environment includes the core services of hosts, open ports, communication connections between hosts, and configuration vulnerability information. In the topology, communication is allowed between neighboring nodes. Each honeypot node is randomly assigned vulnerabilities, along with open ports such as SSH, FTP, RDP, and others to better simulate remote connections and similar services. The environment has independent training termination constraints:

- When a honeypot is exposed, that is, the deception is detected by the attack agent.
- When the cumulative reward for a honeypot agent reaches the threshold α.
- When the last honeypot node is occupied (the value of rewards is not certain).

We designed three experiments as follows based on the AARF framework and with the support of HoneypotSim:

- By comparing with the single attack chain generation method, we verified the effectiveness of the proposed method in generating valid attack chains based on the collection of attack chains. The stability of our method was further verified based on generating attack chains of different lengths.
- Leveraging the classic DQN algorithm, we modeled the interaction between a defense agent and an automated attacker to engage in the game. We analyzed the experiment results to validate the usability and effectiveness of the AARF framework.
- We compared two baseline agent models (both DQN-agent and PPO-agent, respectively) and the DQN with DVLS agent model. We analyzed the comparison results to prove that the dynamic value label sampling method was efficient and reliable.

5.1. Experimental Configuration

The configuration parameters of HoneypotSim can be customized. In the setup of the basic experimental environment, we configured a honeynet environment that included five honeypot nodes. Among them, client1 and client2 belonged to the warning honeypots, GitHubProject and GiteeProject belonged to the trapping honeypots, and Server belonged to the countering honeypot. When an attacker performed malicious actions, AARF-based honeypots enabled rapid and efficient interactive responses. We quantified the anti-identification ability of the honeypot to the probability set (0.1,0.2) according to Equation (2). For the defender reward mechanism, positive feedback rewards non-linearly increased within the range of (5,15) based on Equation (3). We also provided a fixed negative feedback reward of -5 for the failure responses to honeypot nodes.

The simulation environment also contained information related to the attack agents. The attack chain generated by the attack agents was used to represent the attacks in the intranet lateral movement scenario. Specifically, based on the HMM model for the ATT&CK lateral movement phase, it contained a combination of techniques which were executed sequentially to complete the response interaction with the honeypot. The length of the base valid attack chain was five. The core idea of RL-based honeypots is to maximize the cumulative reward through iterative optimization of strategies to achieve the defense response objectives. The tree types of functional honeypots designed were all built on RL algorithms. For honeypots, the purpose of response actions is to establish a highly interactive connection with attackers. The size of the action space was 20, including reasonable response actions designed based on specific attack requests. Partial action classification details are shown in Table 2. The same response action could match multiple attack actions, so it was a many-to-many mapping to the attack action. The state space primarily consisted of discrete state features such as current_defender_state, open_port, file_status, process_status, and so on, representing the observations of the current node regarding the external environment. After a series of iterative training, the agent was allowed to choose response actions at each step based on the current strategy, thereby maximizing the reward within its field of view. Both the attack chain length and the honeynet environment could be dynamically adjusted

for testing and analyzing the performance of the AARF framework and RL in different attack scenarios and honeynet environments.

Table 2. Honeypot response actions classification.

Defender Agent	Action Id	Response Actions
Honeypot	action1	Successfully called service
	action2	Authentication passed
	action3	Login successful
	action4	Content copied successfully
	action5	Obtaining content data
	action6	Accept and click on the phishing link
	action7	Service utilization successful
	action8	The required software already exists
	action9	Software deployment/transfer succeeded
	action10	Obtain remote service permissions

5.2. Experimental Results and Analysis

5.2.1. Validation of Valid Attack Chain Generation

Attack agents have the goal of generating a complete attack chain that is effectively usable. Our proposed method of generating a valid attack chain based on a collection of candidate attack chains can quickly obtain a valid attack chain and thus improve the efficiency of attacks. By simulating honeynet environments with different configurations, we conducted experiments under the scenarios of generating attack chains of 3–8-step actions, respectively. The experimental results are shown in Figure 7. By analyzing the experimental results, we found that in attack chain generation with fewer attack steps, the single attack chain generation method did not differ much from our generation method in terms of generation time. However, as the number of attack steps included in the attack chain increased, the advantage of our method gradually expanded. This is because as the complexity of generating attack chains increased, the number of calls to the HMM for a single attack chain generation method increased significantly, increasing the generation time. Our method selected valid attack chains from the candidate attack chain collection and made partial adjustments, so it could improve the generation efficiency. Overall, our method could output a valid complete attack chain much faster than the single attack chain generation method.

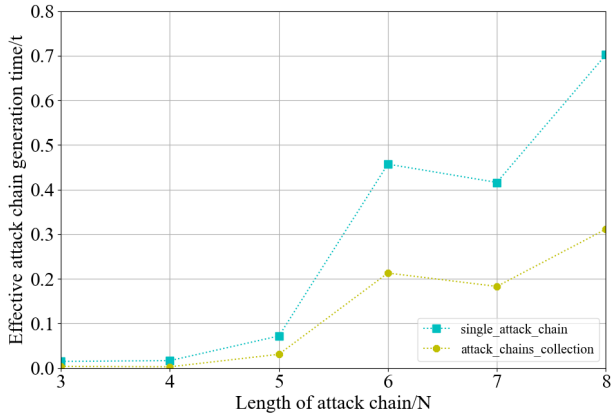


Figure 7. Comparison of valid attack chain generation time.

5.2.2. AARF Framework Feasibility Verification

The AARF framework consists of three parts: the custom honeynet environment, the HMM-based attack agent, and the reinforcement learning-based defense agent. The framework implements an intelligent deception defense response against automated multi-step attacks in a multi-honeypot environment and demonstrates the whole process of interaction

response through visualization. Using the DQN and PPO algorithms, we conducted a feasibility validation of the AARF framework. We built defense agents capable of effectively interacting with the environment. They collaboratively made response decisions against intrusion attacks. Figure 8 shows the attack and defense process from client1 to Server in a visual way. It can be seen that the attacker is guided by the honeypot to access different nodes in turn. Different honeypots complete different defense actions and restore the attack path. This provides an intuitive understanding of interaction games and demonstrates the ability of the AARF framework to automatically respond to the attack chain.

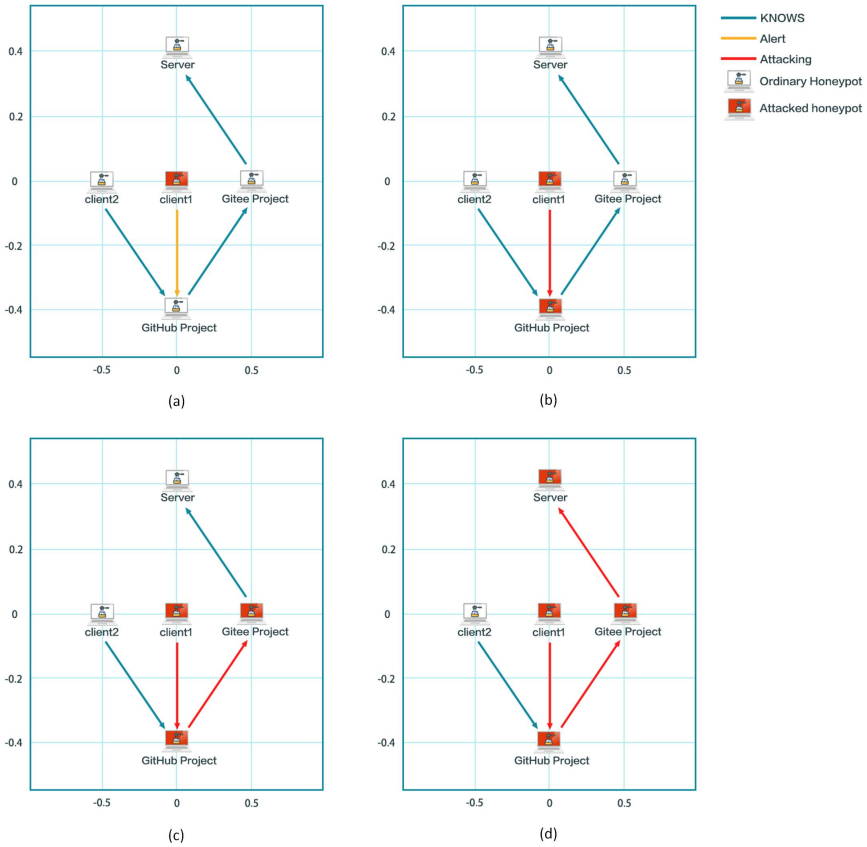
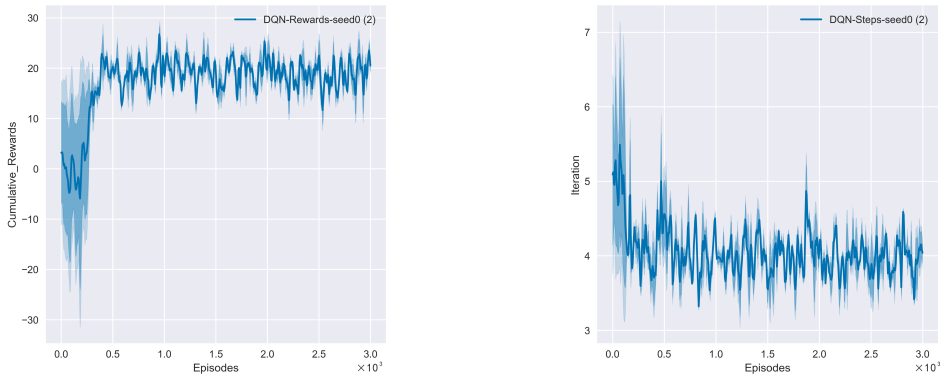


Figure 8. The interaction process of honeypots and attacker. (**a**) Communication alarm; (**b**) threat records-1; (**c**) threat records-2; (**d**) traceability and countermeasures.

Figure 9 quantifies the experimental results of the AARF framework, which shows the cumulative reward figure and episode steps' trend figure of the trained DQN. We focused on the ϵ-greedy strategy, adjusting the action strategy at different stages. In the early stage of training, honeypot agents spent more time to explore the environment using DQN according to AARF with the probability ϵ. As the learning process progressed, the agent gained a more complete understanding of its action space and observation state by learning. It selected the optimal action with the highest possible reward to be fully exploited based on the action distribution within its action space to achieve convergence and stabilization. This demonstrated that the environment configuration supported the reasonable training of agents and has good learning feasibility. It can be seen that both the cumulative reward and the number of iterations converged effectively after a period of training. This means that the agents were able to autonomously learn effective response strategies and apply them in a randomly configured HoneypotSim environment.



(**a**) (**b**)

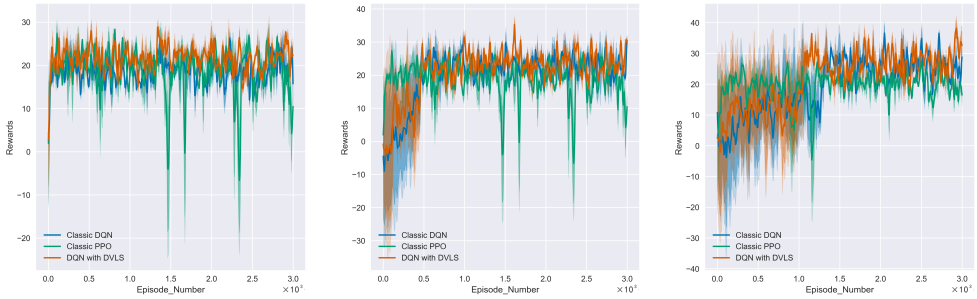
Figure 9. DQN algorithm operating effect based on AARF framework (**a**) DQN cumulative reward; (**b**) DQN iteration round.

5.3. Algorithm Comparative Verification

For the RL algorithm, the initialization settings for relevant parameters were as follows: the discount factor was set to 0.90; the size of the replay buffer was set to 500; the learning rate was set to 0.01; the honeypot's training was based on the TensorFlow platform. The RMSProp optimizer was used to minimize the defined loss during training.

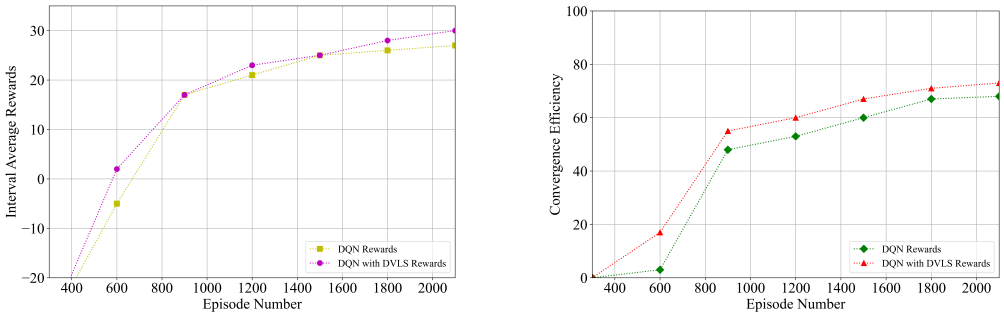
We compared and analyzed three different algorithms in a honeynet scenario containing five honeypots, six honeypots, and seven honeypots of different complexities. This also corresponded to the modeling of attack agents generating attack chains of different lengths. The results are shown in Figure 10. From Figure 10a, we can see that the DQN with DVLS in the early stages had comparable utility to the DQN, which continuously improved and eventually remained stable as the training proceeded. After 1000 episodes, the convergence of all methods tended to stabilize. Compared to the other methods, the DQN with DVLS achieved faster convergence and the highest cumulative reward. From Figure 10a–c, as the complexity of the honeynet environment increased, the convergence of the DQN with DVLS was always the best. This shows the efficiency and stability of the DVLS sampling method, which is attributed to the efficiency of its sampling method. Since the DQN with DVLS labels the value weight of stored data, it allows a more efficient selection of samples for learning. Overall, the PPO algorithm learned more efficiently at the start of training, but it was more volatile as training continued. This may be related to our discrete state-action space environment.

A comparison of the DQN and the DQN with DVLS methods in terms of average cumulative reward and convergence efficiency is shown in Figure 11. Figure 11a,b show the average cumulative rewards of the agents per 300-episode interval and the convergence efficiency of the strategies, respectively. The average cumulative rewards of honeypot agents based on the DQN with DVLS exceeded the DQN by more than eight percent through the calculation, and the convergence efficiency by approximately five percent. The average cumulative reward and convergence efficiency are important metrics for evaluating reinforcement learning models. This indicated that the DQN with DVLS method based on the AARF framework was more efficient in sampling during the training process and could generate the optimal honeypot response strategy with better results faster. Meanwhile, the multi-stage interaction with the attack chain was accomplished based on the generated response strategy, which effectively enhanced the interaction between the honeypot and the attack agent. Thus, it improved the concealment and confusion of the honeypot, extended the behavioral chain of the attack in the honeypot, effectively delayed the attack, captured more of its behavioral characteristics, and further improved the efficiency of deception defense. Therefore, it is of great significance for the research on automated attack response strategies of honeypots.



(**a**) (**b**) (**c**)

Figure 10. Comparisonof algorithm performance in different honeynet environments (**a**) Comparative results in a HoneypotSim environment with 5 honeypots; (**b**) comparative results in a HoneypotSim environment with 6 honeypots; (**c**) comparative results in a HoneypotSim environment with 7 honeypots.



(**a**) (**b**)

Figure 11. Comparison of average reward and convergence efficiency: (**a**) average cumulative reward values; (**b**) average convergence efficiency.

6. Conclusions

In this paper, we focused on effectively enhancing the interaction of honeypots when faced with multi-step sequential attack chains in dynamic games. We made progress in implementing the AARF framework, which consisted of three parts, a honeynet environment, an attack agent, and a defense agent, and could autonomously generate honeypot response strategies for interacting with attack agents. In the HoneypotSim environment, we designed three functional types of honeypot agents to respond collaboratively to an attack chain. For the attack agents, we proposed an automated attack chain generation method based on an HMM combined with the generic threat framework ATT&CK. It was divided into seed attack chain generation, dynamic attack chain evolution, and valid attack chain verification. This effectively improved the efficiency of valid attack chain generation and completed the groundwork for the effective training of honeypots in attack and defense games. Honeypot agents interacted with attack agents in a dynamic honeynet environment based on RL. Considering the problems posed by the random uniform sampling method of RL, we proposed the sampling method of DVLS. The DVLS method labeled the experience samples stored in the experience replay with value labels based on feedback rewards and anti-identification capabilities and guided the agents to learn high-value sample experiences. Through experimental validation, our method could effectively learn high-value sample data, thus accelerating the convergence of the training process and effectively enhancing the interactions of the honeypot.

In future research work, we have the following research plans. Aiming at the attack agents in the AARF framework, we will further consider introducing the vulnerability exploitation methods from the CVE and CNVD vulnerability databases to enrich the

attack method library and generate more diversified attack chains. For the defense agents, based on the reasonable response, we will design and introduce a collaborative defense mechanism to the AARF framework. According to the functional characteristics of different honeypots, we will achieve functional synergy and complete real defense actions such as threat alert, threat alert, traceability countermeasure, etc., to realize active defense. Intelligent honeynet deployment strategies will also be further considered, based on a multi-agent reinforcement learning approach to collaborate on deployment and response tasks. For the virtual honeynet environment, a larger-scale honeynet scenario will be designed to verify the effectiveness of the framework in complex network environments. Finally, future research will involve implementing our proposed AARF framework in real-world multi-honeypot scenarios. This will allow us to better demonstrate the practical value of our research in real-world applications.

Author Contributions: Conceptualization, L.W. and J.D.; methodology, L.W. and J.D.; validation, L.W., H.T. and J.Z.; formal analysis, L.W., J.D. and Y.X.; investigation, L.W., J.D. and Z.Z.; writing—original draft preparation L.W., J.D. and Z.L.; writing—review and editing, L.W., J.D., Z.G. and R.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Guangdong Basic and Applied Basic Research Foundation (2023A1515011698), Guangdong High-level University Foundation Program (SL2022A03J00918), Major Key Project of PCL (PCL2022A03), and National Natural Science Foundation of China (grant no. 62372137).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Togay, C.; Kasif, A.; Catal, C.; Tekinerdogan, B. A firewall policy anomaly detection framework for reliable network security. *IEEE Trans. Reliab.* **2021**, *71*, 339–347. [CrossRef]
2. Zhang, Z.; Wang, L.; Chen, G.; Gu, Z.; Tian, Z.; Du, X.; Guizani, M. STG2P: A two-stage pipeline model for intrusion detection based on improved LightGBM and K-means. *Simul. Model. Pract. Theory* **2022**, *120*, 102614. [CrossRef]
3. Rohith, C.; Kaur, G. A comprehensive study on malware detection and prevention techniques used by anti-virus. In Proceedings of the 2021 2nd International Conference on Intelligent Engineering and Management (Iciem), London, UK, 28–30 April 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 429–434.
4. Liu, S.; Wang, S.; Sun, K. Enhancing Honeypot Fidelity with Real-Time User Behavior Emulation. In Proceedings of the 2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S), Porto, Portugal, 27–30 June 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 146–150.
5. López-Morales, E.; Rubio-Medrano, C.; Doupé, A.; Shoshitaishvili, Y.; Wang, R.; Bao, T.; Ahn, G.J. Honeyplc: A next-generation honeypot for industrial control systems. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, 9–13 November 2020; pp. 279–291.
6. Yamamoto, M.; Kakei, S.; Saito, S. Firmpot: A framework for intelligent-interaction honeypots using firmware of iot devices. In Proceedings of the 2021 Ninth International Symposium on Computing and Networking Workshops (CANDARW), Matsue, Japan, 23–26 November 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 405–411.
7. Team., Microsoft Defender Research CyberBattleSim. 2021. Available online: https://github.com/microsoft/cyberbattlesim (accessed on 10 September 2021).
8. Rahul, S.; Vajrala, C.; Thangaraju, B. A novel method of honeypot inclusive WAF to protect from SQL injection and XSS. In Proceedings of the 2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON), Bengaluru, India, 19–21 November 2021; IEEE: Piscataway, NJ, USA, 2021; Volume 1, pp. 135–140.
9. Erdődi, L.; Sommervoll, Å.Å.; Zennaro, F.M. Simulating SQL injection vulnerability exploitation using Q-learning reinforcement learning agents. *J. Inf. Secur. Appl.* **2021**, *61*, 102903. [CrossRef]
10. Del Verme, M.; Sommervoll, Å.Å.; Erdődi, L.; Totaro, S.; Zennaro, F.M. Sql injections and reinforcement learning: An empirical evaluation of the role of action structure. In Proceedings of the Secure IT Systems: 26th Nordic Conference, NordSec 2021, Virtual Event, 29–30 November 2021; Proceedings 26; Springer: Berlin/Heidelberg, Germany, 2021; pp. 95–113.
11. Foley, M.; Maffeis, S. HAXSS: Hierarchical reinforcement learning for XSS payload generation. In Proceedings of the 2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Wuhan, China, 9–11 December 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 147–158.
12. Baillie, C.; Standen, M.; Schwartz, J.; Docking, M.; Bowman, D.; Kim, J. Cyborg: An autonomous cyber operations research gym. *arXiv* **2020**, arXiv:2002.10667.

13. Straub, J. Modeling attack, defense and threat trees and the cyber kill chain, att&ck and stride frameworks as blackboard architecture networks. In Proceedings of the 2020 IEEE International Conference on Smart Cloud (SmartCloud), Washington, DC, USA, 6–8 November 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 148–153.
14. Siniosoglou, I.; Efstathopoulos, G.; Pliatsios, D.; Moscholios, I.D.; Sarigiannidis, A.; Sakellari, G.; Loukas, G.; Sarigiannidis, P. NeuralPot: An industrial honeypot implementation based on deep neural networks. In Proceedings of the 2020 IEEE Symposium on Computers and Communications (ISCC), Rennes, France, 7–10 July 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–7.
15. Mfogo, V.S.; Zemkoho, A.; Njilla, L.; Nkenlifack, M.; Kamhoua, C. AIIPot: Adaptive intelligent-interaction honeypot for IoT devices. In Proceedings of the 2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Toronto, ON, Canada, 5–8 September 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 1–6.
16. Huang, L.; Zhu, Q. Adaptive honeypot engagement through reinforcement learning of semi-markov decision processes. In Proceedings of the Decision and Game Theory for Security: 10th International Conference, GameSec 2019, Stockholm, Sweden, 30 October–1 November 2019; Proceedings 10; Springer: Berlin/Heidelberg, Germany, 2019; pp. 196–216.
17. Pauna, A.; Iacob, A.C.; Bica, I. Qrassh-a self-adaptive ssh honeypot driven by q-learning. In Proceedings of the 2018 International Conference on Communications (COMM), Bucharest, Romania, 14–16 June 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 441–446.
18. Cabral, W.; Valli, C.; Sikos, L.; Wakeling, S. Review and analysis of cowrie artefacts and their potential to be used deceptively. In Proceedings of the 2019 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 5–7 December 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 166–171.
19. Kristyanto, M.A.; Studiawan, H.; Pratomo, B.A. Evaluation of Reinforcement Learning Algorithm on SSH Honeypot. In Proceedings of the 2022 6th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE), Yogyakarta, Indonesia, 13–14 December 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 346–350.
20. Liu, Q.; Stokes, J.W.; Mead, R.; Burrell, T.; Hellen, I.; Lambert, J.; Marochko, A.; Cui, W. Latte: Large-scale lateral movement detection. In Proceedings of the MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM), Los Angeles, CA, USA, 29–31 October 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6.
21. Bowman, B.; Laprade, C.; Ji, Y.; Huang, H.H. Detecting lateral movement in enterprise computer networks with unsupervised graph {AI}. In Proceedings of the 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020), San Sebastian, Spain, 14–15 October 2020; pp. 257–268.
22. Ho, G.; Dhiman, M.; Akhawe, D.; Paxson, V.; Savage, S.; Voelker, G.M.; Wagner, D. Hopper: Modeling and detecting lateral movement. In Proceedings of the 30th USENIX Security Symposium (USENIX Security 21), Vancouver, BC, Canada, 11–13 August 2021; pp. 3093–3110.
23. Strom, B.E.; Applebaum, A.; Miller, D.P.; Nickels, K.C.; Pennington, A.G.; Thomas, C.B. Mitre att&ck: Design and philosophy. In *Technical Report*; The MITRE Corporation: McLean, VA, USA, 2018.
24. Amal, M.; Venkadesh, P. Review of cyber attack detection: Honeypot system. *Webology* **2022**, *19*, 5497–5514.
25. Pandire, P.A.; Gaikwad, V.B. Attack detection in cloud virtual environment and prevention using honeypot. In Proceedings of the 2018 International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 11–12 July 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 515–520.
26. Shrivastava, R.K.; Bashir, B.; Hota, C. Attack detection and forensics using honeypot in IoT environment. In Proceedings of the Distributed Computing and Internet Technology: 15th International Conference, ICDCIT 2019, Bhubaneswar, India, 10–13 January 2019; Proceedings 15; Springer: Berlin/Heidelberg, Germany, 2019; pp. 402–409.
27. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv* **2013**, arXiv:1312.5602.