Honeypot Intrusion Detection System using an Adversarial Reinforcement Learning for Industrial Control Networks

Abbasgholi Pashaei[1], Mohammad Esmaeil Akbari[2*], Mina Zolfy Lighvan[3], Asghar Charmin[4]
1,2,4- Department of Electrical Engineering, Ahar Branch, Islamic Azad University, Ahar, Iran.
Email: a-pashaei@iau-ahar.ac.ir, m-akbari@iau-ahar.ac.ir (Corresponding author), a_charmin@sut.ac.ir.
3- Department of Electrical and Computer Engineering Faculty, Tabriz University, Tabriz, Iran.
Email: mzolfy@tabrizu.ac.ir

Received: 26 October 2022 Revised: 14 November 2022 Accepted: 25 December 2022

ABSTRACT:
Distributed Denial of Service (DDoS) attacks are a significant threat, especially for the Internet of Things (IoT). One approach that is practically used to protect the network against DDoS attacks is the honeypot. This study proposes a new adversarial Deep Reinforcement Learning (DRL) model that can deliver better performance using experiences gained from the environment. Further regulation of the agent's behavior is made with an adversarial goal. In such an environment, an attempt is made to increase the difficulty level of predictions deliberately. In this technique, the simulated environment acts as a second agent against the primary environment. To evaluate the performance of the proposed method, we compare it with two well-known types of DDoS attacks, including NetBIOS and LDAP. Our modeling overcomes the previous models in terms of weight accuracy criteria (> 0.98) and F-score (> 0.97). The proposed adversarial RL model can be especially suitable for highly unbalanced datasets. Another advantage of our modeling is that there is no need to segregate the reward function.

KEYWORDS: Intrusion Detection, Honeypot, Markov Decision Process, Adversarial Learning.

1. INTRODUCTION

Distributed Denial of Service (DDoS) attacks cause flood traffic from multiple sources to selected nodes and impede the flow of legal information across a network [1]. They are still a significant threat, especially to the Internet of Things (IoT). If the victim node is a server that needs to process information quickly, the entire network operation will be stopped. One approach practically used to protect the network against DDoS attacks is the honeypot [2]. The goal is to examine a potential attacker's behavior by analyzing it. In this way, we can strengthen the primary system based on the patterns learned by the honeypot so that it is less vulnerable to similar attacks in the future. The first line of defense is to detect DDoS attacks. Then, attack flows can be identified and labeled in real-time in the second line of defense.

Recently, the importance of Intrusion Detection Systems (IDSs) has been highlighted due to their economic gain to organizations [3, 4]. Recently, the attention of Machine Learning (ML) researchers has been focused on the use of IDS as a Decision Support System (DSS) [5]. In this regard, it is tried to design IDSs with the lowest detection error and high prediction speed. Security and privacy concerns are increasingly disrupting access to operational data. Using suitable ML methods for Honeypot systems can speed up the detection of attacks. While existing technologies can effectively support Honeypot, they lack comprehensive activity validation. Unfortunately, one of the most critical weaknesses of ML-based honeypots is that they cannot detect network changes effectively. Another major challenge in designing IDSs is the issue of feature extraction and preprocessing [6, 7, 8]. In other words, associating features and intrusion labels is not an easy task at all. The actual penetration mode is challenging. For this reason, in almost all studies in this field, supervised learning methods are used for feature generation [9, 10].

Reinforcement Learning (RL) methods are powerful tools for improving the learning of supervised learning methods [11, 12]. The application of Deep Reinforcement Learning (DRL) [13, 14] has grown dramatically. The main distinguishing feature of RL methods is the use of experiences gained from the environment over time [15]. This is precisely the way humans and animals learn. The environment takes action from the agent and returns new modes and

Paper type: Research paper
DOI: 10.30486/mjtd.2023.1977229.1011
How to cite this paper: A. Pashaei, M. Esmaeil Akbari, M Zolfy Lighvan and A. Charmin, "**Honeypot Intrusion Detection System using an Adversarial Reinforcement Learning for Industrial Control Networks***", Majlesi Journal of Telecommunication Devices*, Vol. 12, No. 1, pp. 17-28, 2023.

rewards [16]. Then, based on the acquired states and rewards, the algorithm optimizes the agent's policy function. The agent uses the policy function to calculate the following best action based on the current state and the rewards provided by the environment [17].

Most of the studies conducted so far on IDSs have been using ML techniques. Among these, the most considerable contribution of studies is for classical ML techniques [16, 18, 19, 20]. Also, several studies have been conducted to evaluate ML methods, such as RL [11, 12] and DRL [3, 21]. Some recent research has used the manipulation of specific models of ML, called adversarial machine learning [16, 22, 23, 24]. The most important feature of this type of study is injecting insufficient data into the ML algorithm to fail the classification model. Each of these studies has been modeled differently. The most significant differences are the design of the states, actions, and reward functions. Thanks to powerful reward functions, DRL is very suitable for online training and can quickly respond to changes in network conditions.

This paper proposes a high-accuracy DRL model combining simulated and real-world environments. If the agent's prediction is correct, it will be given a positive reward from the environment. We define the objective function as equal to the reward of the environment. The goal is to maximize the objective function. The main strength of the proposed modeling is that it randomly extracts new samples (states) from the pre-registered sample dataset. The proposed system, using the experiences gained from the environment, can produce better results contrary to the policy of the adversarial agent. If the classifier's mispredictions increase, then environmental behavior actively reduces the rewards given to the agent. This way, the agent is forced to learn the most complex cases. Our primary motivation in this research is to design two-way honeypots using the RL approach. It allows two-way honeypot surveillance of internal/external attackers. These honeypot devices can be used in industrial LANs in the cyber-physical IoT. It can be a legal way to provide different types of data to domestic and foreign consumers. The proposed RL-based system can strengthen the security of honeypots against all kinds of attacks. It detects internal/external attacks through multilevel RL techniques. In the proposed method, the event management technique is based on the safe distribution of toxins. The system develops DQ, Adversarial Environment (AE), and RL modules for successful attack detection at runtime.

We used the CICDDoS2019 (Knowledge Data Discovery) Database [25, 26] from the Canadian Institute for Cyber Security in previous research. This dataset contains attacks that may compromise IDS security and make it suitable for analysis. To evaluate the performance of the proposed method, we compare it to two categories of DDoS attacks, including NetBIOS and LDAP. NetBIOS stands for Network Basic Input/Output System [27]. Its primary purpose is to allow applications in a LAN to communicate and create sessions to access shared resources. Also, LDAP stands for Lightweight Directory Access Protocol [28]. It is an industrial protocol for controlling distributed directory information on a network.

The most important contributions of this paper are as follows:

- We introduce a hybrid architecture of supervised and RL models to predict attack patterns. It can be generalized to almost all other types of industrial networks. The proposed model using adversarial RL can be especially suitable for highly unbalanced datasets because samples matched with incorrect classification labels are often used for training.
- We show through experimentation that the proposed new model has a higher prediction accuracy than nonlinear models and, at the same time, requires much less prediction time.
- The next step is to generate a data packet for testing and transmit it to the "patient." The tester package is identical to any other data package. However, since its contents are obscured and filled with a random data stream, the victim may not deduce that it came from the honeypot.

In the rest of this paper, we have a complete study of the previous studies in Section 2; Section 3 presents the formulation of the problem; Section 4 presents the experiments to evaluate the modeling. Finally, in Section 5, the conclusion is presented.

2. RELATED WORK

Most of the studies that have been done so far on intrusion detection systems have focused on ML methods. Since the scope of this paper is reinforcement learning, our primary focus is on reviewing these types of studies. A review of the literature shows that much of previous research has focused on improving the performance of methods. Most of the research has included comparing the performance of the proposed strategies with other Machine Learning (ML) approaches. Also, numerous studies have been conducted to evaluate ML methods such as Deep Reinforcement Learning (DRL) [3, 21].

Pashaei et al. [20] used SARSA based Markov Decision Process (MDP) reinforcing learning method to identify attacks. They compared the DRL-based strategy with a conventional scheduling-based plan. This evaluation was done considering learning periods, implementation time, and rewards.

In another study, Holgado et al. [18] proposed an ML technique for detecting multi-stage attacks that use a Hidden Markov Model (HMM) to predict the

attacker's following action. In [19], a comprehensive study of DRL cybersecurity capabilities is conducted, including real-time traces and simulations. They focused on mastering adversarial reinforcement learning techniques and, at the same time, examining the effect of multi-agent DRL models on intrusion detection systems. Al-Amin et al. [29] proposed an online deception technique that includes various scenarios. In these scenarios, defending activities dynamically affect attacking plans and tactics. Defender actions are modeled as a Partially Observable Markov Chain (POMDP). This RL model monitors the defender's belief in the attacker's progress. As a result, the defender distracts the attacker from the decoy nodes.

Alavizadeh et al. [5] proposed a DRL model to identify and classify different types of network intrusion attacks. Their proposed model uses a labeled dataset as input. It can learn automatically by leveraging trial and error techniques. The most critical weakness of their method is how the MDP adapts to the environment. Unfortunately, this study does not explain how to adapt to the environment. Therefore, it is impossible to make a clear judgment about the learning rate of the proposed method based on environmental experiences. This, in turn, obscures the system's accuracy in detecting network attacks.

Some studies [11, 12] have used a combination of reinforcement learning (RL) and game theory. In addition, machine learning may be used to examine players 'beliefs to develop a strategy and competitors' beliefs to predict their movements. [30] provides a context-based intrusion detection system that uses a distributed network of independent DRL agents to improve detection accuracy for threats. They use the NSL-KDD, UNSWNB15, and AWID datasets to achieve greater accuracy and lower the false-positive rate (FPR). Another study [31] used RL as a sensor network analysis engine and IDS. Combining adaptive machine learning with clustered IDS results in higher accuracy and better recall rates.

It is possible to manipulate specific models of ML. This is called adversarial machine learning. It is a malicious attempt to enter insufficient data into the ML algorithm to cause the classification model to fail. Adversarial is often used in two ways: poisoning the environment and avoidance. The first goal is manipulating the training data to produce the wrong classification decision. On the other hand, the second technique assumes that the model is already trained and tries to correct its behavior to make incorrect predictions. Numerous studies have examined the effect of poor learning problems on NIDS [24].

In other studies, RL has been used to detect anomalies. Caminero et al. [22] advised a multi-agent RL system. Their proposed model has higher accuracy and F-score than the famous ML algorithms. Suwannalai et al. [23] presented multi-agent RL with DRL.

Adversarial reinforcement learning [22] is a type of RL that combines environmental learning with an adaptive RL algorithm. Establishing a reliable reward mechanism is of great importance. The SMOTE [16] is a machine learning technique that solves problems using an unbalanced dataset. Unbalanced datasets often occur in practice and need to be controlled. They solve this problem by developing an adversarial RL with SMOTE leverage. This research was performed on the NSL-KDD dataset and showed a remarkable superiority over previous DRL methods. Unfortunately, this technique cannot distinguish unknown classes with high accuracy only by relying on positive samples. The authors in [32] modified the sampling function of the instruction data. This method provides incentives to detect errors during training by sampling the training data. The proposed technique performs best when the DDQN algorithm is used. It leads to an excellent F-score in AWID and NSL-KDD datasets.

In [33], an optimal strategy for deceptive resources was achieved by leveraging an attacker-defender game. In [4], DRL combined with ML is used to detect DoS attacks on the server side. The main weakness of the proposed method is that it can only be used for single DoS attack detection. Dowling et al. [34] conducted another study on deploying a honeypot using the RL. One of the significant drawbacks of this study is that it uses only the SSH protocol and does not support interaction-level samples.

So far, several RL strategies have been used for Intrusion Detection Systems (IDSs), which aim to improve accuracy. Unfortunately, in this research, few explanations are provided regarding the formulation of RL or the precise setting of the above parameters. They also do not explain learning and interacting with the infrastructure network environment. Contrary to the above research, in this article, we provide a complete formulation of development and implementation for the next Honeypot iteration based on the DRL technique. The data gained from these studies was very beneficial in building the proposed EIDS- AERL system. Adversarial Environment Reinforcement Learning (AERL) based Honeypot was used as a substitute strategy for standard EIDS techniques in the suggested method. AERL learns by interacting with two agents and obtaining a reward function. However, AERL performs better in the unbalanced domain.

3. PROPOSED METHOD

This study aims to develop a new multi-agent adversarial RL strategy based on the SARSA deep algorithm [17] to improve the classification of different types of minority attacks. As mentioned earlier, these tests run on the CICDDoS2019 dataset [25, 26]. This

research enhances the approach of [35] and makes essential modifications to it that lead to improved performance. Readers interested in studying AE can refer to [22]. Details of the original AE-DQN can also be found in [23]. The generalities of the RL algorithm are also found in [36].

We use Q-learning in the family of Finite Markov Decision Process (FMDP) approaches to model the environment. Here, "Q" refers to the function that the algorithm calculates. This function also called a *reward function*, is the expected reward r_{t+1}, for an action a_t performed in a given state s_t. Agents learn to treat the environment based on previous experiences they have gained. Q-value measures the value of each experience. Also, the reward function tunes the Q-value. As shown in Fig. 1, after each action a_t, the agent is moved from the state s_t to a new state s_{t+1} and receives a reward r_{t+1}. This reward reflects the importance of the action and can be the basis for deciding what to do next. The sequences of states in which each agent enters over time can be represented by $(s_0, r_0, a_0, s_1, r_1, a_1, s_2, r_2, a_2, ..., s_t, r_t, a_t)$.

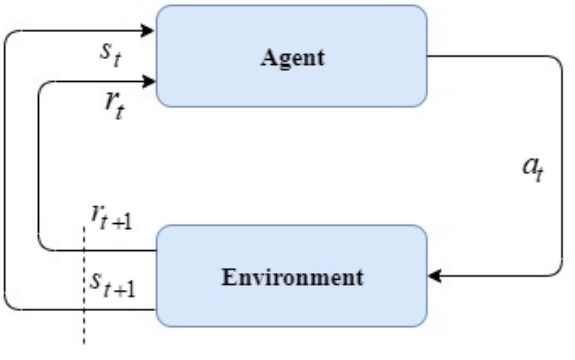

Fig. 1. Environmental interactions in Q-learning [17].

Q-learning is called model-free because it does not need prior knowledge of the environment to learn the value of an action in a given state. It can solve our problem with rewards without the need for consistency. Here, an optimal policy can maximize the expected value of the total reward through multiple iterations. In this way, Q-training can find the optimal action-selection policy for each FMDP over time through a somewhat random policy. The components of our RL-based offloading approach are as follows:

Agent: In our modeling, IDS is an agent.

State: Let $s_t \in \mathbb{S}$ is the current state of the agent, which $\mathbb{S}$ denotes the state space. Thus, we have

$$\mathbb{S} = \{"BENIGN", "Attack"\}. \quad (1)$$

Action: We represent the current action by $a_t \in \mathbb{A}$, which $\mathbb{A}$ denotes the action space. Thus, we have

$$\mathbb{A} = \begin{Bmatrix} "QV_i = Predict\ (Current-state)" \\ , \\ "AV_i = argmax\ (QVi)" \end{Bmatrix} \quad (2)$$

,where action vector (AV) as $AV_i = Rnd\ (0,1)$ and Q vectors (QV_i) for all $\forall_i \in b_s$. Here, b_s is the space of random actions. Here, actions are fed into AV_i. Also, 0, 1, and 2 represent the BENIGN, LDAP, and NetBIOS , respectively. There are many ways to choose a strategy. We use the greedy technique, in which the action with the highest value is superior to the others. This action is represented as $a_t \doteq \arg\max_a Q_t(s_t, a)$.

Transition Function: It is the likelihood of performing the action $a_t = a$ to transmit the agent from a state $s_t = s$ to a state $s_{t+1} = s'$. We represent this probability with $P(s', r \mid s, a)$, which $P : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \to [0,1]$ and $\sum_{s'} \sum_{r} P(s', r \mid s, a) = 1$.

Reward Function: After the agent chooses the desired action and moves to a new state, it receives a reward r_{t+1}^i.

Q-learning can be shown as $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$. In other words, starting from the state s_t, it acts a_t, receives the reward r_{t+1}, and finally reaches the new state s_{t+1}. In the new state, it aims to perform an action a_{t+1}. These operations result in updating $Q(s_t, a_t)$.

An agent receives information from the environment and responds with action in an RL pattern. This action ultimately changes the environment. For the actions the agent performs, a reward is given by the environment depending on the effectiveness.

The essential corrections we make to the above research are as follows:

- We create an artificial ecology using a labeled attack dataset.
- The agent creates a classification using simulated environment modes to predict honeypot labels.
- The simulated environment rewards the agent for correct/incorrect predictions.

The upgraded framework supports a well-known RL method called *Q-learning* [17] to identify honeypots using data. The Q-learning algorithm finds

the best Q function for the agent (here, classifier). The Q function evaluates each state-action pair. Over time, with the agent choosing the current policy, this amount equals the total reward.

The Q-learning algorithm is one of the off-policy RL methods. We adopt the simplest form of it, which is called single-step Q-learning. It is defined as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \begin{bmatrix} r_{t+1} \\ + \\ \gamma \max_b Q(s_{t+1}, b) \\ - \\ Q(s_t, a_t) \end{bmatrix} \quad (3)$$

In the above formula, Q is the learned value-action function. Also, α is the learning rate. Another important hyperparameter is the discount factor $0 \le \gamma \le 1$. When it is small, the agent will have slower actions.

After determining the transition function P and the received reward r_{t+1} by the agent, the MDP problem can be easily solved using dynamic programming algorithms. Here, the core idea is to use the value function $V(s)$ to find the optimal action a_*. The optimal action in any state s_t is the action that brings the most reward to the agent. For this purpose, the state value function must be expressed in the following form, which is known as the Bellman equation:

$$V_*(s) = \max \mathrm{E}(r_{t+1} + \gamma . V_*(s_{t+1}) \mid s_t = s,\ a_t = a) \quad (4)$$

The above formula after simplification can be rewritten as follows:

$$V_*(s) = \max_a \sum_{s',r} P(s', r \mid s, a)[r + \gamma . V_*(s')] \quad (5)$$

One of the most common ways to solve the Bellman equation is to rewrite it in the following recursive form:

$$V_{t+1}(s) = \max_a \sum_{s',r} P(s', r \mid s, a)[r + \gamma . V_t(s')] \quad (6)$$

Thus, the value of all states can be obtained. Given that $0 \le \gamma \le 1$, it can be proved that the Bellman equation will converge and, therefore, the solution is as follows:

$$V_*(s) = \lim_{t \to \infty} V_t(s) \quad (7)$$

The key advantage of Q-learning over other RL methods is that it converges very quickly. The main reason is that Q can directly approximate the optimal action-value function, q_*. Here, the policy determines which state-action pairs to visit and update. Like most RL methods, the prerequisite for convergence is that all pairs continue to be updated. Under this assumption and other common indefinite approximation conditions in the sequence of size-step parameters, it is found that Q converges with a probability of 1 to q_*. Another advantage of Q-learning is that it does not require a specific environment model. In RL terminology, it is called a model-free method and does not require a predetermined policy to find any optimal state-action pair. The pseudo-code of the Q-learning for the offloading problem is shown in *Algorithm 1*.

Algorithm 1 Pseudo-code of the Q-learning for the problem.

Input: the set of states $\mathcal{S}$, the set of actions $\mathcal{A}$, the set of terminal states $\mathcal{T}$

Output: the optimal action-value function vector $\mathbf{Q}_*$

Set $t = 0$
for each agent ***do***
  for $s_t \in \mathcal{S}$ ***do***
    for $a_t \in \mathcal{A}$ ***do***
      Initialize $Q(s_t, a_t)$ arbitrarily
      Initialize terminal state value, $Q(\mathcal{T}, .) = 0$
    end for
  end for
end for
repeat
  Initialize $\mathbf{S}_t$
  repeat (for each step of the episode)
    Choose $\mathbf{A}_t$ from $\mathbf{S}_t$ using policy derived from $\mathbf{Q}_t$ (e.g., Ɛ-greedy)
    Take action $\mathbf{A}_t$, observe $\mathbf{R}_{t+1}$, $\mathbf{S}_{t+1}$
    $\mathbf{Q}(\mathbf{S}_t, \mathbf{A}_t) \leftarrow \mathbf{Q}(\mathbf{S}_t, \mathbf{A}_t) + \alpha \left[\mathbf{R}_{t+1} + \gamma \max_{\mathbf{B}} \mathbf{Q}(\mathbf{S}_{t+1}, \mathbf{B}) - \mathbf{Q}(\mathbf{S}_t, \mathbf{A}_t) \right]$ using Eq. (3)
    $\mathbf{S}_t \leftarrow \mathbf{S}_{t+1}$

17: ***until*** $\mathbf{S}_t \in \mathcal{T}$
18: $t \leftarrow t+1$
19: ***until*** *Max_Num_Episodes*
20: $\mathbf{Q}_* \leftarrow \mathbf{Q}(\mathbf{S}_t, \mathbf{A}_t)$ using Eq. (7)
21: return $\mathbf{Q}_*$

We use a Neural Network (NN) with fully-connected nodes to approximate the Q function. Here, we use the features extracted from the labeled dataset as the input to the NN. The NN output shows the Q function for a set of possible actions. As mentioned earlier, the Q function indicates how beneficial the current action is. The pseudo-code of Deep Q-Network (DQN) has been shown in *Algorithm 2*. Instead of the expression used in Eq. (3), the DQN model uses a quadratic loss function to update the Q function. This loss function is defined as follows:

$$Q(s_t, a_t) \leftarrow \left[r_{t+1} + \gamma \max_b Q(s_{t+1}, b) - Q(s_t, a_t) \right]^2 \quad (8)$$

Adversarial RL works very well for unbalanced datasets. We attach a new RL agent to the environment to capture this feature. The training operation for environmental and main classification agents is performed parallel using DQN. The action (output) of the classifier predicts the type of influence for the sample dataset (input). The environmental agent action will be the attack class used to generate new samples in the training process. In other words, the environmental agent tries to direct the generation of samples to increase the errors produced by the classification agent. This reduces the reward for the classifier and forces it to focus on the most challenging samples. All positive rewards for the manager will be negative for the environment. In this way, the environment learns which classes produce the most failures for the primary agent. It will try to increase the number of such states with a certain probability. To do this, we use two Q functions: a *(s, a)* function responsible for optimizing the classifier and another *(s, a)* function for optimizing the environment. Using the CICDDos2019 dataset, the primary Q function has a set of actions (a_c [0-2]) proportional to the number of classifier classes. The environmental Q function, on the other hand, contains a set of measures (a_e-train [0-6] or a_e-test [0-11]). These actions correspond to potential attacks on the training/test dataset.

The strategy adopted throughout the training corresponds to a diminishing $\varepsilon - greedy$, ensuring that the exploration begins with a high value and gradually diminishes as the episodes go. Throughout the dataset, we treat each episode as a training cycle. An episode is also often referred to as an *epoch*. The agent and the environment determine their behaviors based on their policy, with each sample having a distinct lower constraint ε. The lower bound ε is kept low for each classifier to optimize detection. Conversely, the lower bound of the environmental policy is set as a hyper-parameter.

Regarding the difference between how our proposed method works with DQN, pay attention to the following steps:

- The environmental agent and the classifier agent randomly initialize the Q-value. In addition, an initial state s_0 is randomly generated from the state of the dataset to activate the Q function.
- The environmental agent chooses a honeypot action a_{e_t} based on its current policy and states.
- The environmental agent randomly chooses the current state from the set s_t inside the feature-label pair (s_t, a_{e_t}).
- After the environmental agent takes a certain state, the counterparty agent, similar to DQN, performs the classification based on the policy and maps it to a_{e_t}.
- After comparing the environmental action with the actual label, if the two are different from each other, the appropriate reward is given to the agent by the environment.
- Similar to DQN, the environmental agent is provided with a new state relevant to the next feature-label pair $(s_{t+1}, a_{e_{t+1}})$.
- Using the obtained reward values and predicted future states, the policy functions of the classifying agents and the environment agents are modified according to the DQN update rule.

The initial value of the reward function is 0.99, with a positive reward of +0.99 and a negative reward of 0. Cross-entropy and stratified hinge functions are used to calculate the distance between expected and actual actions (labels) and how that distance relates to the reward. The shorter the distance, the greater the reward. Another possible reward function is to offer a variety of rewards depending on the attack. Finally, the most superficial reward of 0.99 is selected for superior performance. The steps of the method adapted from a standard DQN algorithm are shown in *Algorithm 2*. These different stages enable the environmental factor

to be trained with an adversarial strategy.

In the proposed algorithm, we have made three other modifications to DQN as follows:

- The final method is based on DDQN, a version of DQN that uses two separate networks to select and evaluate actions.
- The actual loss function used to train agents is Huber, similar to a quadratic loss up to a certain threshold. The Huber loss seeks to minimize the importance of slopes that may exhibit explosive behavior.
- Because our agents are made using two separate neural networks, we use two $\varepsilon - greedy$ methods. We start both networks significantly, decreasing to 0.73 for the environment network and 0.02 for the classifier network. Experimentally, we have discovered that high ε values are critical for the environment (active exploration) to achieve excellent results.

Algorithm 2 Pseudo-code of AE-RL for the problem.

1: $Q_c(s, a_{c_t})$, $Q_e(s, a_{e_t})$

2: For episode

3: $s_0 = A$ *random sampling* CICDDos2019 dataset

4: Initialize Environment Agent: a_{e_t} using $\varepsilon - greedy$ derived from $Q_e(s, a_{e_t})$

5: Replace s_t where the label is a_{e_t}

6: Initialize Classifier Agent: a_{c_t} using $\varepsilon - greedy$ derived from $Q_c(s_t, a_{c_t})$

7: $RL \rightarrow (r_{c_t}, r_{e_t}, s_{t+1}) \rightarrow rewards\ for\ environment \rightarrow (r_{c_t}, r_{e_t})$

8: Environment Agent: $a_{e_{t+1}}$ using $\varepsilon - greedy$ derived from $Q_e(s_t, a_{e_t})$

9: Replace s_{t+1} where the label is $a_{e_{t+1}}$

10: Classifier Agent: $a_{c_{t+1}}$ using $\varepsilon - greedy$ derived from $Q_c(s_t, a_{c_t})$

11: Q function update: gradient descent on the loss function

12: $(r_{e_t} + \gamma_{a_{e_{t+1}}} \max Q_e(s_{t+1}, a_{e_{t+1}}) - Q_e(s_t, a_{e_t}))^2$

13: $(r_{c_t} + \gamma_{a_{c_{t+1}}} \max Q_c(s_{t+1}, a_{c_{t+1}}) - Q_c(s_t, a_{c_t}))^2$

Fig. 2 shows the structure of the proposed method. The upper part of the figure shows the training step, while s_0 represents the initial states randomly selected from the dataset. Also, the prediction step is shown in the lower part of the figure, which relies entirely on the previously trained classifier. Adversarial (environment) and defense agent (classifier) are complex neural networks. The architecture consists of a multilayer neural network with 100 neurons per layer in both samples. While the complexity of this design offers a competitive reaction time, RL training optimizes weights and biases. Both attackers and AEs may use the current sample properties as input while generating network-based outputs. This classifier produces one of eleven possible attack types as the honeypot output. As a result, the defense agent neural network creates a classifier. However, the adversarial agent's neural network can not be considered a classifier but a generator of all attacks.

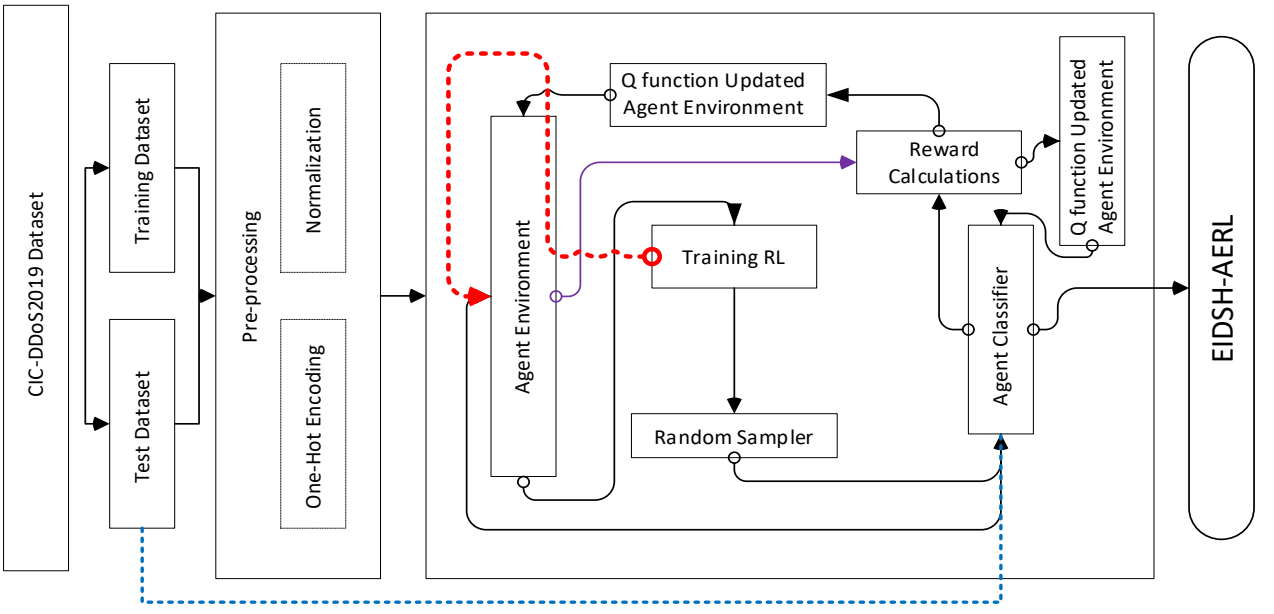

Fig. 2. The structure of the proposed method.

4. PERFORMANCE EVALUATION

We implemented our proposed model in Python using the *TensorFlow* package on the CICDDoS2019 dataset [25, 26]. The CICDDoS2019 dataset includes several samples containing network features and related labels for honeypots with varying potential values , such as binary anomalies or multi-classes. The CICDDoS2019 dataset results from a collaboration between the Canadian Communications Security Establishment (CSE) and the Canadian Institute for Cyber Security (CIC). It includes two different DDoS attacks: NetBios and LDAP. These features were scaled to the [0, 1] range. The final post-preprocessing dataset has 164 attributes. Each test and training sample offers one of 11 possible items. To help interpret the data, the training/test tags fall into three new categories: BENIGN, NETBIOS, and LDAP. Except for BENIGN, which means no infiltration, all previous classes are related to attacks.

We trained the model using 80% of the obtained data and used the remaining 20% as a validation set to fine-tune the meta-parameters. This study labeled network features as states and values as actions. Apart from the input and output layers, we used a total of two hidden layers with a RELU activation function in a fully connected neural network. Various important parameters must be discovered and examined during training to identify the most appropriate and optimal model values. At the beginning of the training, the discount rate ε is set to 0.1. It shows that the agent performs the exploration with unpredictability and a deception rate of 0.98. It decides how the agent's learning performance will improve in response to future rewards.

We chose a discount factor γ = [0, 1] and the size of the batch as1. The amount of loss/rewards obtained during the training process is shown in Fig. 3 for different values of discount factors γ for target prediction. Fig. 3-a and 3-b show the reward and loss values by setting them to 0.001. We gradually increased the learning rate 0.2 and calculated the loss/reward values in Figs. 3-a and 3-b. Findings show that increasing the discount coefficient γ increases the value of losses. As can be seen from the figure, the loss value in the worst-case scenario reaches 1.5 when the discount factor is 0.001. As shown in Fig. 3, discount factor values reflect the rapid growth trend in the reward values. Based on the behavior of the adversary agent, a lower discount rate γ leads to a lower loss value. This, in turn, leads to more accurate model learning, especially when the episode counter is small.

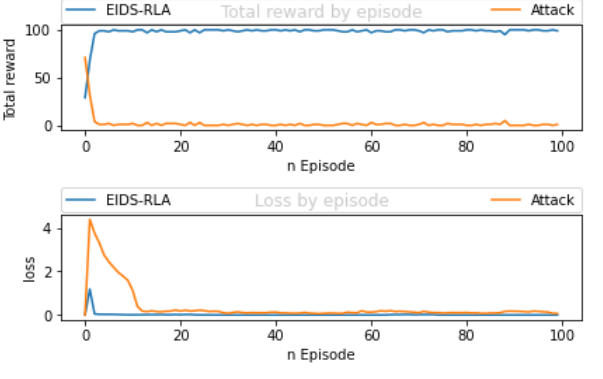

Fig. 3. Loss/reward values of the learning process in the proposed model.

Fig. 4 shows a histogram of the attacks generated by an attacker during training in different epochs (iterations). Histograms show only three special episodes on the training dataset. At the beginning (period zero), the environment randomly generates attacks. As the environmental agent's training progresses, it learns which attacks maximize rewards. As can be seen from the figure, the frequency of different attacks has a random distribution. However, the figure shows an upward trend in specific attacks, namely NETBIOS and LDAP, while at the same time, the usual traffic is decreasing. This behavior is expected to see in a dynamic (intelligent) algorithm. Simply speaking, the proposed algorithm tries to regulate the bias caused by the variable dataset during the training phase. Comparing the frequency of attack types in the training dataset with the distribution of attacks created by the environmental agent in the lower right part of Fig. 4 has an interesting implication. It shows how the intelligent environment actively changes the unbalanced data distribution to increase classification performance.

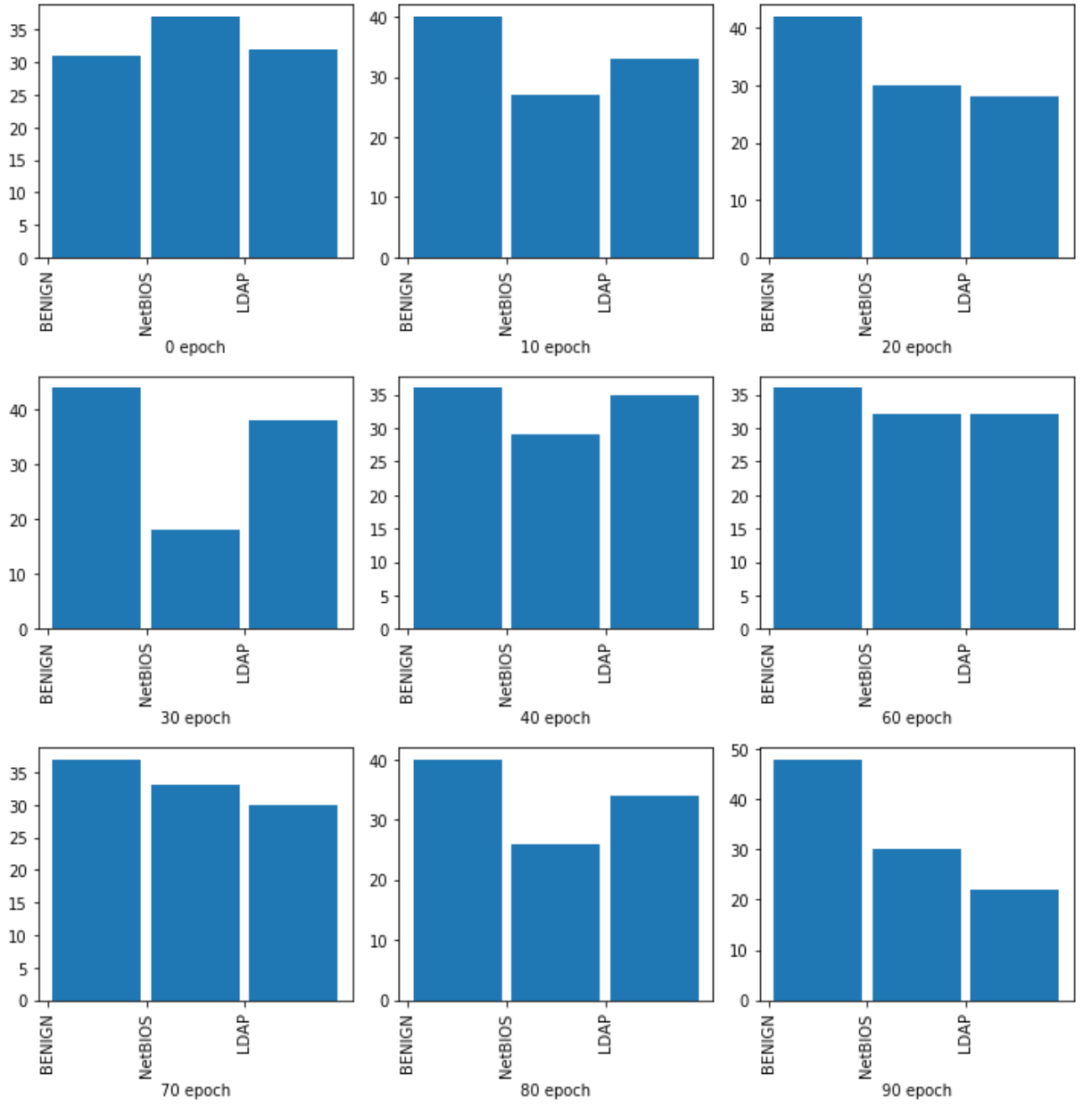

Fig. 4. Distribution of attacks generated by an attacker during training in different epochs.

Fig. 5 shows the confusion matrix of the proposed model to create the policy network. As is shown, the proposed method produces minor false negatives for identity fraud and flooding attacks. It tries to increase the classification of unusual classes. As seen from the figure, the proposed method may minimize the false-negative rate for regular classes to obtain a significant false-positive rate for the standard classes. Thus, minimizing false-negative rates for duplicate classes improves the classification of less common ones. We observe that the proposed method dramatically increases the frequency of accurate estimates for the LDAP minority class.

Overall, it can be concluded that the proposed method has a high prediction accuracy for normal samples. This is because it clusters the majority samples to the closest minority samples. At the same time, since the majority classes are clustered and do not cover a wide range of locations, more predictions are made for the minority classes, leading to a significant number of accurate estimates. On the other hand, the proposed method works well in most BENIGN classes, albeit with a lower proportion of valid estimates. This is because it retains the majority samples closest to the minority samples. Simply speaking, the proposed method has more accurate predictions for the majority classes and, at the same time, fewer predictions for the minority classes. It works well in identifying BENIGN, NETBIOS, and LDAP threats. In addition, it works well in identifying BENIGN samples despite their small size.

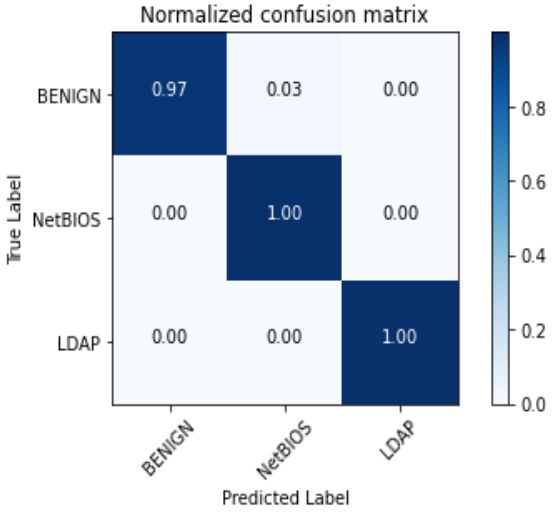

Fig. 5. Confusion matrix of the proposed model.

As shown in Fig. 6, the proposed method generates accurate values for different labels (BENIGN, NETBIOS, and LDAP). At the same time, it offers the highest F-scores and accuracy, especially for minority groups (BENIGN).

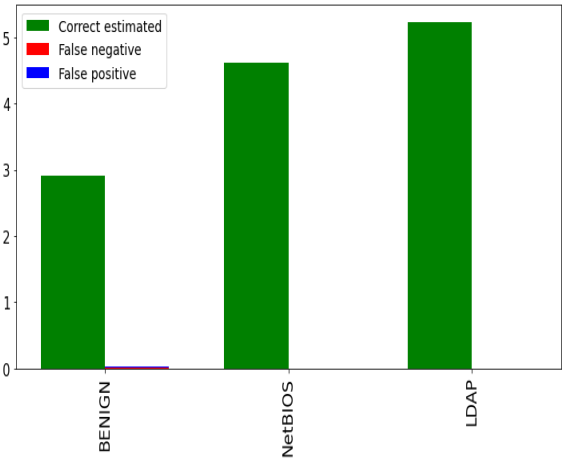

Fig. 6.F-score values of the proposed model.

One of the most important tricks of our proposed model is that it creates a significant decrease in the number of desirable false negatives in exchange for a slight increase in the number of false positives. The following points summarize our findings:

- The classification results with the proposed method are entirely comparable to state-of-the-art techniques' results.
- Since the proposed method requires less training and much shorter forecasting time, it is suitable for use in practical honeypots in typical IoT environments.
- The proposed method can reduce classification errors in classes that occur less frequently.
- The only drawback of the proposed method is its false negative. False negatives are often found in less regular classes because algorithms maximize overall prediction performance. Fortunately, in a real-world honeypot environment, this phenomenon rarely occurs.

5. CONCLUSION

This paper presented an adversarial Deep Reinforcement Learning (DRL) model for honeypots. It can do better by using experiences derived from the environment. Further regulation of the agent's behavior is made with an adversarial goal. In the proposed model, the environment tries to complicate the difficulty level of predictions. As the second agent, we used a simulated environment. In addition, a new method for oversampling low-performance categories was introduced, which helps train the AERL algorithm.

We evaluated the performance of our proposed model with NetBIOS and LDAP attacks on the CICDDoS2019 dataset. The evaluation results showed that the proposed method requires less training and a much shorter forecasting time. Therefore, it is suitable for practical honeypots in typical IoT environments. It can also reduce classification errors in classes that occur less frequently. Our proposed model is not only efficient for highly imbalanced datasets but also does not require

discretization of the reward function. The only drawback of the proposed method is its false negative. Fortunately, in a real-world honeypot environment, this situation rarely occurs.

Other RL methods can be considered for future research, especially Markov Decision Process (MDP). Also, combining RL with metaheuristic techniques can probably have a significant effect on speeding up the algorithm.

Compliance with ethical standards

Informed consent: Informed consent was obtained from all participants included in the study.

Conflict of interest On behalf of all authors, the corresponding author states no conflict of interest.

Funding The authors did not receive support from any organization for the submitted work.

Data Availability The datasets generated during and analyzed during the current study are available in the [data.mendeley.com/datasets/gxntkyyjvf/1] repository, [DOI: 10.17632/gxntkyyjvf.1].

REFERENCES

[1] Tian, W., Du, M., Ji, X., Liu, G., Dai, Y. and Han, Z., 2021. **"Honeypot detection strategy against advanced persistent threats in industrial internet of things: a prospect theoretic game".** *IEEE Internet of Things Journal*, 8(24), pp.17372-17381.

[2] Harikrishnan, V., Sanket, H.S., Sahazeer, K.S., Vinay, S. and Honnavalli, P.B., 2022. **"Mitigation of DDoS Attacks Using Honeypot and Firewall".** *In Proceedings of Data Analytics and Management* (pp. 625-635). Springer, Singapore.

[3] K. Sethi, Y. V. Madhav, R. Kumar, and P. Bera, **"Attention based multiagent intrusion detection systems using reinforcement learning,"** *Journal of Information Security and Applications*, vol. 61, p. 102923, 2021.

[4] Lopez-Martin, M., Carro, B., & Sanchez-Esguevillas, A. (2020). **"Application of deep reinforcement learning to intrusion detection for supervised problems".** *Expert Systems with Applications*, 141, 112963.

[5] Alavizadeh, H., Jang-Jaccard, J., & Alavizadeh, H. (2021). "**Deep Q-Learning based Reinforcement Learning Approach for Network Intrusion Detection".** *arXiv preprint arXiv*:2111.13978.

[6] A. Pashaei, M. E. Akbari, M. Zolfy Lighvan, and A. Charmin4, **"A Honeypot-assisted Industrial Control System to Detect Replication Attacks on Wireless Sensor Networks",** *Majlesi Journal of Telecommunication Devices*, Vol. 11, No. 3, pp. 155-160, 2022.

[7] Yang, Z., Liu, X., Li, T., Wu, D., Wang, J., Zhao, Y. and Han, H., 2022. **"A systematic literature review of methods and datasets for anomaly-based network intrusion detection".** *Computers & Security*, p.102675.

[8] Imran, M., Haider, N., Shoaib, M. and Razzak, I., 2022. **"An intelligent and efficient network intrusion detection system using deep learning".** *Computers & Electrical Engineering*, 99, p.107764.

[9] Roy, S., Li, J., Choi, B.J. and Bai, Y., 2022. **"A lightweight supervised intrusion detection mechanism for IoT networks"**. *Future Generation Computer Systems*, 127, pp.276-285.

[10] Teixeira, D., Malta, S. and Pinto, P., 2022. **"A Vote-Based Architecture to Generate Classified Datasets and Improve Performance of Intrusion Detection Systems Based on Supervised Learning".** *Future Internet*, 14(3), p.72.

[11] Y. Liu, H. Wang, M. Peng, J. Guan, J. Xu, and Y. Wang, "**DeePGA: A privacy-preserving data aggregation game in crowdsensing via deep reinforcement learning,"** *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4113–4127, 2020.

[12] Q. Xu, Z. Su, and R. Lu, **"Game theory and reinforcement learning based secure edge caching in mobile social networks,"** *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3415–3429, 2020.

[13] Gupta, G.P., 2022. **"Intrusion Detection Framework Using an Improved Deep Reinforcement Learning Technique for IoT Network".** *In Soft Computing for Security Applications* (pp. 765-779). Springer, Singapore.

[14] Praveena, V., Vijayaraj, A., Chinnasamy, P., Ali, I., Alroobaea, R., Alyahyan, S.Y. and Raza, M.A., 2022. **"Optimal Deep Reinforcement Learning for Intrusion Detection in UAVs".** *CMC-COMPUTERS MATERIALS & CONTINUA*, 70(2), pp.2639-2653.

[15] Naghdehforoushha, M., Dehghan Takht Fooladi, M., Rezvani, M.H., Gilanian Sadeghi, M.M., 2022, **"BLMDP: A New Bi-level Markov Decision Process Approach to Joint Bidding and Task-Scheduling in Cloud Spot Market",** *Turk J Elec Eng & Comp Sci*, DOI: 10.3906/elk-2108-89.

[16] Ma, X., & Shi, W. (2020). "**Aesmote: Adversarial reinforcement learning with smote for anomaly detection".** *IEEE Transactions on Network Science and Engineering*, 8(2), 943-956.

[17] Sutton RS**,"Barto AG. Reinforcement learning: An introduction"**. *MIT press*; 2018 Nov 13.

[18] P. Holgado, V. A. Villagrá, and L. Vazquez, **"Real-time multistep attack prediction based on hidden markov models,"** *IEEE Transactions on Dependable and Secure Computing*, 2017.

[19] T. T. Nguyen and V. J. Reddi, "**Deep reinforcement learning for cyber security,"** *arXiv preprint arXiv*:1906.05799, 2019.

[20] Pashaei, A., Akbari, M. E., Lighvan, M. Z., & Charmin, A. **"Early Intrusion Detection System**

using honeypot for industrial control networks". *Results in Engineering*, 100576. (2022).

[21] B. Hu and J. Li, **"Shifting deep reinforcement learning algorithm towards training directly in transient real-world environment: A case study in powertrain control,"** *IEEE Transactions on Industrial Informatics*, 2021.

[22] Caminero, G., Lopez-Martin, M., &Carro, B. **"Adversarial environment reinforcement learning algorithm for intrusion detection".** *Computer Networks*, Vol. 159, 2019, pp. 96–109. doi: 10.1016/j.comnet.2019.05.013.

[23] Suwannalai, Ekachai, and Chantri Polprasert. **"Network Intrusion Detection Systems Using Adversarial Reinforcement Learning with Deep Q-network",** *18th International Conference on ICT and Knowledge Engineering (ICT&KE)*, 2020, IEEE.

[24] PACHECO, Yulexis et SUN, **"Weiqing. Adversarial Machine Learning: A Comparative Study on Contemporary Intrusion Detection Datasets",** *ICISSP*, 2021. pp. 160-171.

[25] Ferrag MA, Shu L, Djallel H, Choo KK. "**Deep learning-based intrusion detection for distributed denial of service attack in Agriculture 4.0"**. *Electronics. 2021 Jan*;10(11):1257.

[26] Sharafaldin I, Lashkari AH, Hakak S, Ghorbani AA. **"Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy".** *In 2019 International Carnahan Conference on Security Technology (ICCST)* 2019 Oct 1 (pp. 1-8). IEEE.

[27] Hussain, Y.S., 2020**. "Network Intrusion Detection for Distributed Denial-of-Service (DDoS) Attacks using Machine Learning Classification Techniques".**

[28] Kshirsagar, D. and Kumar, S., 2022. **"A feature reduction based reflected and exploited DDoS attacks detection system".** *Journal of Ambient Intelligence and Humanized Computing*, 13(1), pp.393-405.

[29] M. A. R. Al Amin, S. Shetty, L. Njilla, D. K. Tosh, and C. Kamhoua, **"Online cyber deception system using partially observable Monte-Carlo planning framework,"** *in Proceedings of the International Conference on Security and Privacy in Communication Systems. Springer*, 2019, pp. 205–223.

[30] K. Sethi, E. S. Rupesh, R. Kumar, P. Bera and Y. V. Madhav, **"A contextaware robust intrusion detection system: a reinforcement learning-based approach,"** *International Journal of Information Security*, 2019.

[31] S. Otoum, B. Kantarci and H. Mouftah, **"Empowering Reinforcement Learning on Big Sensed Data for Intrusion Detection,"** *in Proceedings of IEEE International Conference on Communications (ICC)*, 2019.

[32] Veluchamy, S., & Kathavarayan, R. S. (2021). **"Deep reinforcement learning for building honeypots against runtime DoS attack".** *International Journal of Intelligent Systems*.

[33] S. Wang, Q. Pei, J. Wang, G. Tang, Y. Zhang, and X. Liu, **"An intelligent deployment policy for deception resources based on reinforcement learning,"** *IEEE Access*, vol. 8, pp. 35 792–35 804, 2020.

[34] Dowling, S., Schukat, M., & Barrett, E. (2018, September). **"Using reinforcement learning to conceal honeypot functionality".** *In Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 341-355). Springer, Cham.

[35] Dang QV, Vo TH. **"Reinforcement learning for the problem of detecting intrusion in a computer system"**. *In Proceedings of Sixth International Congress on Information and Communication Technology 2022* (pp. 755-762). Springer, Singapore.

[36] Zhao D, Wang H, Shao K, Zhu Y. **"Deep reinforcement learning with experience replay based on SARSA**". *In2016 IEEE Symposium Series on Computational Intelligence (SSCI)* 2016 Dec 6 (pp. 1-6). IEEE.