IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p-ISSN: 2278-8727, Volume 27, Issue 4, Ser. 1 (Jul. – Aug. 2025), PP 01-09
www.iosrjournals.org

Reinforced Honeypot Technique Integrated with Game Theory and Reinforcement Learning (RL) To Enhance Cyber Defence Mechanisms

[1*]Asogwa T.C., [1]Onah C.B.
[1,1]Computer Science Department, Enugu State University of Science and Technology*
Email: [1]tochukwu.asogwa@esut.edu.ng,[1]onahchukwuebuka7@gmail.com*

Abstract
Advanced Persistent Threats (APTs) pose significant risks to modern network infrastructures. Traditional honeypot systems, while useful for deception and attack intelligence gathering, often lack adaptability, making them ineffective against sophisticated adversaries. This study proposes a reinforced honeypot technique that integrates game theory and reinforcement learning (RL) to enhance cyber defence mechanisms. The methodology used for the study is a behaviour-driven development approach. The method follows a structured approach, beginning with the development of a network infrastructure model to simulate real-world attack scenarios. A reinforcement learning algorithm was trained to adapt the honeypot's deception strategies dynamically, ensuring continuous improvement in detecting and misleading attackers. These models were integrated into a reinforced honeypot system, implemented using the Python programming language, and tested in a controlled cybersecurity environment. The results demonstrated that the adaptive honeypot was able to learn and take appropriate action, which modelled exact user behaviour with a normal Q-value of 1. In addition, it was observed that consistently, the model recorded an average of 0.975 accuracy in correctly interacting with the network environment and classifying user actions. The result showed that the model was able to successfully divert the attacker to the decoy facility, where threat intelligence was collected and applied to update the insecurity. This study contributes to cybersecurity by introducing a self-learning, strategically deceptive honeypot framework that enhances cyber threat intelligence gathering and proactive defence strategies. Future research should focus on deep reinforcement learning, cloud-based honeypots, and multi-vector attack detection to further improve honeypot effectiveness in diverse cybersecurity environments.

Keywords: *Advanced Persistent Threats; Honeypot; Game Theory; Reinforcement Learning; Cyber Defence*

Date of Submission: 25-06-2025 Date of Acceptance: 05-07-2025

I. INTRODUCTION

Cyber threats encompass a spectrum of potential events that pose risks to information systems through unauthorised access. Each year witnesses a proliferation of novel cyber threats targeting critical assets across industrial, governmental, and personal networks. Moreover, these threats continually evolve, spawning various iterations with enhanced malicious capabilities, rendering them increasingly intricate and elusive. A prime example is the emergence of the Mirai botnet in 2016, comprising a legion of bots under adversarial control capable of orchestrating Distributed Denial of Service (DDoS) assaults on Internet of Things (IoT) devices (Papaspirou et al., 2021). Rather than resting on their laurels, adversaries have iterated on Mirai, spawning offspring like Persirai to bolster their nefarious activities while evading detection. Security assessments reveal a doubling of botnets following Mirai's debut (Khoje, 2023).

Honeypots fulfil three primary functions: Detection, Prevention, and Research. Notably, their detection capability excels due to a minimal rate of false positives, as legitimate users typically refrain from interacting with them. This attribute grants honeypots a distinct advantage in identifying zero-day attacks compared to conventional security tools. In terms of prevention, honeypots serve as a proactive deterrent, dissuading adversaries through the perception of risk and uncertainty, thus fortifying network defences against potential breaches. Honeypots assume a pivotal role in cybersecurity research by amassing comprehensive data on adversaries' actions and responses (Amal and Venkadesh, 2022). This trove of information proves indispensable for researchers to dissect and understand patterns in adversarial behaviour, offering invaluable insights into the ever-changing tactics and strategies of adversaries. Such not only informs the development of more robust security tools but also aids in proactively identifying and mitigating emerging threats, thereby contributing to a more resilient and proactive cybersecurity landscape (Zhang et al., 2023)

As cyber threats continue to evolve in complexity and sophistication, there is a growing need for innovative cybersecurity technologies to protect sensitive data and infrastructure. One approach gaining traction is the use of deceptive techniques, such as honeypots, which lure attackers into controlled environments to gather intelligence and mitigate potential threats. However, there remains a gap in understanding the effectiveness and limitations of deceptive cybersecurity technologies like honeypots. This study is focused on honeypots and their deception techniques. While several surveys exist in this domain, including recent works by (IIg et al., 2023; Zuzcak and Zenka, 2020; Zhang et al., 2023), none delve into honeynets' deception techniques. This study presents the application of an intelligent honeypot approach, which integrates game theory and reinforcement learning for the enhancement of cyber defence mechanisms.

II. RESEARCH METHODOLOGY

The methodology for this work is the Behavioural Driven Development (BDD) approach. BDD is particularly suitable for this cybersecurity study because it focuses on defining system behaviour from the attacker's perspective, ensuring that the honeypot is designed to handle real-world threats dynamically. Traditional software development methods often emphasise system functionality without explicitly modelling threat interactions, but BDD allows for attack-driven development, where expected cyber-attacks (such as SQL injection, brute-force login attempts, and privilege escalation) are scripted, tested, and refined iteratively. By using natural language descriptions (Gherkin syntax) to define attack-response scenarios, BDD ensures that the honeypot's deception strategies, logging mechanisms, and adaptive responses are systematically validated. Furthermore, BDD supports automated testing and continuous integration, making it possible to refine the honeypot based on evolving attack tactics detected through reinforcement learning. This approach enhances threat intelligence collection, system adaptability, and the overall effectiveness of cybersecurity defences by ensuring that the honeypot not only captures attacker behaviour but also evolves its deception strategies in response to new threats.

1. PRODUCING THE NETWORK ENVIRONMENT FOR HONEYPOT

In achieving this objective, the aim is to create a decoy network environment which looks similar to the real network but has several decoy vulnerabilities. The network decoy network environment was developed with an adaptive honeypot, while the decoy vulnerabilities are deployed with honeytoken (Lee and Park, 2024).
Let the Real system(S_r) represent the actual network, Honeypots (H_i), which constitute the decoy system made of $H_{1,}H_2, \dots\dots\dots\dots\dots\dots\dots H_n$ to mimic the real network. Then the honeynetH_{net}, which is a collection of several honeypots. The honeynet presents the attacker with multiple decoy vulnerabilities, while the honeytokens are placed inside the honeypot to model vulnerabilities. To make these honeypots adaptive and hard for attackers to detect, Equation 1 was used to model the honeypot as a highly interactive adaptive system, considering the state of the honeypot $\sigma_i(t),$ attack vector $A_i(t)$ and defence strategy $D_i(t)$ at H_i at time t.

$$\sigma_i(t+1) = f\big(\sigma_i(t), A_i(t), D_i(t)\big) \tag{1}$$

These adaptive honeypot models can switch between multiple decoy configurations σ_i to mimic real-world vulnerabilities. To model the interaction between the attacker and the adaptive honeypot, a Markov decision model was applied in Equation 2.

$$P\,(\sigma_i(t+1)|\sigma_i(t), A_i(t), D_i(t)) = \pi(\sigma_i(t+1)) \tag{2}$$

Where $\pi(\sigma_i(t+1))$ is the probability of transition to a new state of the honeypot to ensure that attackers continuously face difficulty in differentiating a real and decoy facility.

1.1 To train a Reinforcement Learning Algorithm and Generate a Model for Adaptive Honeypot

The reinforcement learning algorithm used for this work is the Deep Q-Learning Network (DQN) technique. This utilised neural network (Sochima et al., 2025) to approximate the Q-value function, and handle dynamic and continuous state spaces like the network environments, thus making it suitable for adaptive honeypot. Figure 1 presents the components of the DQN.

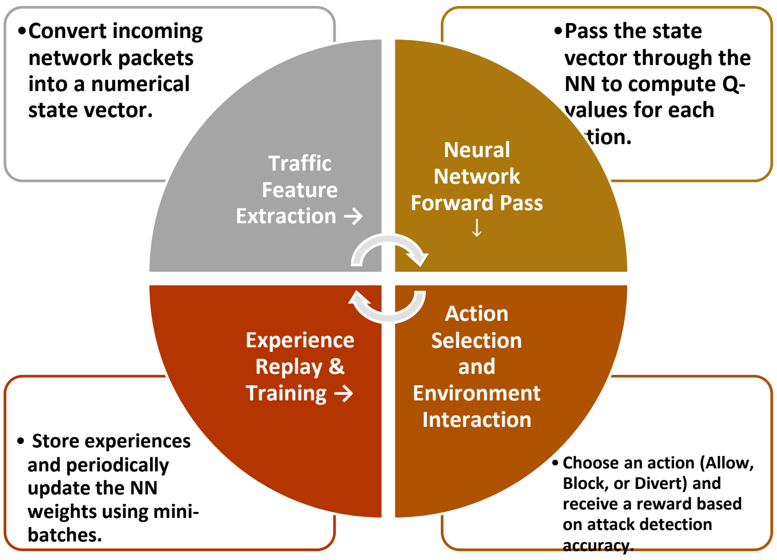

•Convert incoming network packets into a numerical state vector.
•Pass the state vector through the NN to compute Q-values for each action.
Traffic Feature Extraction →
Neural Network Forward Pass ↓
Experience Replay & Training →
Action Selection and Environment Interaction
• Store experiences and periodically update the NN weights using mini-batches.
• Choose an action (Allow, Block, or Divert) and receive a reward based on attack detection accuracy.

Figure 1: Components of the DQN Workflow

Traffic Feature Extraction: The honeypot system captures incoming network traffic, extracting key features such as packet source, destination, protocol type, request rate, and anomaly indicators. These extracted features form the state representation for the DQN model, enabling it to distinguish between normal and malicious traffic patterns.
Neural Network Forward Pass: The extracted traffic features are fed into the deep neural network (DQN model), which processes the input through multiple layers (Ebere et al., 2025). The model assigns a Q-value to each possible action (allow, block, divert), estimating the long-term reward of taking that action in the current state. The neural network architecture is presented in Table 1.

Table 1: The Neural Network Architecture

Component	Description
Input Layer	Accepts **12 extracted features** from network packets
Hidden Layers	Two **fully connected (dense) layers,** each with **128 neurons,** using **ReLU activation**
Output Layer	Three neurons (one for each possible action: **Allow, Block, Divert**) using **Softmax activation** to determine the best action.
Optimization	Uses **Adam optimiser** to update network weights based on Q-value differences.
Loss Function	**Mean Squared Error (MSE)**between predicted and target Q-values**.**
Learning Process	**Experience Replay:** Stores previous state-action-reward transitions and samples them to stabilise learning.

Action Selection and Environment Interaction: Based on the predicted Q-values, the system either chooses the best action (exploitation) or randomly explores new actions (exploration). The selected action is then applied to the network, influencing how traffic is handled, whether attackers are redirected to a decoy, blocked, or mistakenly allowed through.
Experience Replay & Training: To improve decision-making, past experiences (state, action, reward, next state) are stored in a memory buffer. During training, the DQN samples batches of past experiences, using Q-learning to update its weights and minimise errors in action-value predictions, making future decisions more accurate and adaptive. The deep Q-learning Neural Network (DQN) algorithm is presented as follows;

The DQN Algorithm
1. Initialise the DQN components
2. Set random weights as Q(s, a, θ)
3. Initialise the target Q-network $Q^i\ (s, a, \theta)$
4. Initialise replay memory to store experiments (s, a, r, s^i)
5. Observed the state of the environment (s)
6. Select the action using ϵ-greedy policy
7. Execute action a and observe the new state and reward
8. Store outcome in memory
9. Compute target Q-values as $y = r + \gamma \underset{a^i}{max} Q^i\ (s^i\ , a^i\ ;\ \theta^{-1})$
10. Where γ is the discount factor.

```
11. Update Q-network by minimizing loss $L(\theta) = E\,|(y - Q(s,a,\theta))^2|$
12. Update target Q-network
13. End
```

1.2 To integrate the models as a reinforced honeypot against a persistent threat

The Deep Q-Network (DQN) integration into the honeypot was done by transforming the honeypot into an adaptive learning agent that continuously improves its attack detection and diversion capabilities. Unlike traditional honeypots, which rely on static rules, the DQN model enables the system to dynamically classify, react, and adapt based on real-time network traffic behaviour. This was achieved through a reinforcement learning framework, where the honeypot learns from interactions with different types of attacks and adjusts its responses accordingly.

The integration process began by defining the honeypot's environment, which consists of incoming network traffic containing normal requests, reconnaissance attempts, and different types of cyberattacks (SQL Injection, DDoS, Brute Force). A feature extraction module was implemented to analyse key traffic attributes such as packet rate, protocol type, request patterns, and source IP behaviour. These extracted features serve as the state representation for the DQN model, allowing it to recognise the nature of incoming requests.

The DQN agent was then trained using a state-action-reward system, where the state represents the extracted traffic features, the action determines how the honeypot responds (e.g., allow, block, divert to decoy), and the reward function evaluates the effectiveness of the action. If an attack is correctly identified and diverted, the agent receives a positive reward; if a normal request is misclassified, a penalty is applied. The DQN's neural network was trained using past attack logs and real-time traffic data, allowing it to develop an optimal policy for handling different attack scenarios.

Once deployed, the DQN-powered honeypot continuously refines its decision-making through live training. As more attack data is processed, the model updates its Q-values, improving its classification accuracy. This enables the honeypot to intelligently respond to novel and evolving cyber threats, ensuring that attackers are effectively trapped in the decoy system while normal users experience minimal disruptions. By integrating DQN-based learning, the honeypot network becomes a self-adapting defence mechanism, capable of proactive threat mitigation with high accuracy and resilience against sophisticated adversarial techniques.

1.3 The New Network Environment Model with reinforced honeypot

The network environment under study consists of a reinforcement learning-driven honeypot system, integrated with a Deep Q-Network (DQN) to enhance cybersecurity defences. The model is designed to dynamically detect and respond to various cyber threats while ensuring minimal disruption to legitimate network traffic. The environment is structured to include both attackers and normal users, with the honeypot acting as an intelligent security layer that determines whether incoming traffic should be allowed, blocked, or diverted to a decoy system.
The network topology includes three main components:

1. Legitimate Server: The main system where normal traffic should be directed.
2. Adaptive Honeypot: A deception-based security system controlled by a DQN model that detects, classifies, and responds to incoming requests.
3. Attack Sources: Malicious entities attempting various cyber-attacks such as SQL Injection, DDoS, Brute Force, Reconnaissance, and XSS.

The incoming network traffic is monitored in real-time, and the honeypot extracts key features from each packet, such as source IP, request frequency, packet size, protocol type, and payload structure. These features serve as the input state to the DQN model, which then decides the best response action:

- Allow traffic to the legitimate server (if recognised as normal).
- Block traffic (if identified as a high-risk threat).
- Divert traffic to the honeypot for further engagement and monitoring

The environment operates in a reinforcement learning framework, where the DQN model continuously learns from attack patterns and adapts its decision-making over time. The reward function ensures the model prioritises accurate threat detection while reducing false positives. By simulating multiple attack scenarios, the model fine-tunes its policy to efficiently engage attackers, gather intelligence, and protect network assets.

1.4 To implement the model using the Python programming language

The implementation of the adaptive honeypot security model was carried out in Python, leveraging Deep Q-Network (DQN) for reinforcement learning-based threat detection and response. The model was built using key libraries such as TensorFlow/Keras for deep learning, OpenAI Gym for reinforcement learning simulation, Scikit-learn for feature extraction, and PyShark for real-time packet capture. The honeypot system was developed using Scapy for packet analysis and Flask to simulate an interactive decoy environment. Attack datasets, including real-world SQL Injection, DDoS, and Brute Force logs, were used to train the DQN model for robust threat detection.

The training process involved defining the reinforcement learning environment, where network packets were preprocessed into feature vectors representing different attack types. The DQN agent was designed with a neural network that maps network traffic states to optimal defensive actions (Allow, Block, or Divert). The agent was trained using the experience replay technique, where past attack interactions were stored in a memory buffer and sampled for training to improve learning stability. Epsilon-greedy exploration was applied to balance exploration (discovering new strategies) and exploitation (using learned knowledge). The reward function was defined such that correct attack detection and mitigation earned positive rewards, while false positives and misclassifications resulted in penalties.

Once trained, the model was deployed into a live network monitoring system where it analysed incoming packets in real-time. Suspicious traffic was automatically diverted to a high-interaction honeypot, logging attacker behaviour while protecting the main server. The system was tested against multiple cyberattacks, including SQL Injection, DDoS, and Reconnaissance, to evaluate its effectiveness.

3.5 Testing Attack Vectors and Their Impacts

To test the new network environment, several attack vectors such as SQL injection, denial of service, man in the middle, phishing and brute force were all used to test the network.

i. SQL Injection Attack

In this test, we simulated SQL Injection attempts where attackers tried to manipulate database queries by injecting malicious SQL code. The DQN-based honeypot identified abnormal query structures and redirected attackers to a decoy database. Over time, the model learned to detect new patterns of SQL injection, improving its false positive rate and minimising legitimate query blocks.

ii. Distributed Denial-of-Service (DDoS) Attack

We launched a simulated DDoS attack by generating high-volume traffic to flood the network. The honeypot classified abnormal traffic spikes and diverted malicious packets to a controlled decoy system. The DQN improved its response time with each training iteration, reducing network downtime and enhancing resource allocation.

iii. **Brute Force Attack**

A brute force attack was tested by repeatedly attempting password guesses on authentication systems. The DQN learned patterns of repeated login failures and adapted its response by applying rate-limiting mechanisms and directing attackers to a fake login portal. This reduced system overload and prevented unauthorised access.

iv. **Phishing-Based Exploitation**

We tested social engineering-based attacks where attackers attempted to trick users into revealing credentials. The honeypot intercepted these phishing attempts and used reinforcement learning to classify and block suspicious email patterns. Over time, the system improved its ability to differentiate phishing attempts from normal user activity.

v. **Man-in-the-Middle (MITM) Attack**

We simulated an MITM attack where an attacker intercepted data between a user and the server. The honeypot detected anomalies in SSL/TLS handshakes and redirected suspected attackers to a fake communication channel. The DQN model improved its ability to detect unauthorised interceptions, protecting user data integrity.

2. SYSTEM IMPLEMENTATION RESULTS

This section presents the result of the training to learn the network environment and then adapt to help improve decision-making and detection of threats on the network facility. During the training of the DQN, different episodes are points where the model is evaluated to determine its ability to learn and adapt to the environment. Training of the DQN monitors the model performance in learning and interacting with the network environment under different conditions to know when to block, divert or allow access of user packets to the network. Figure 3 presents the training performance considering different results of the model at each episode.

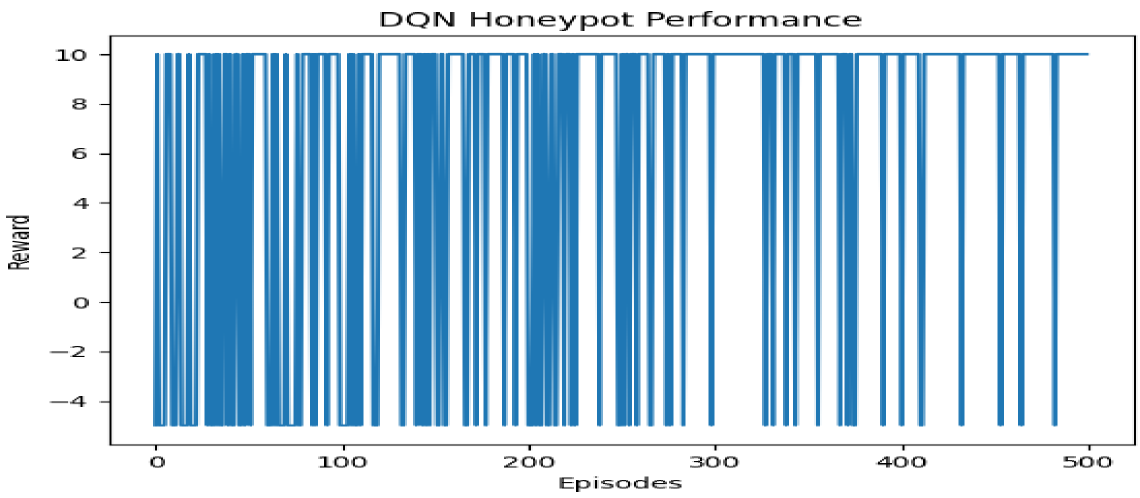
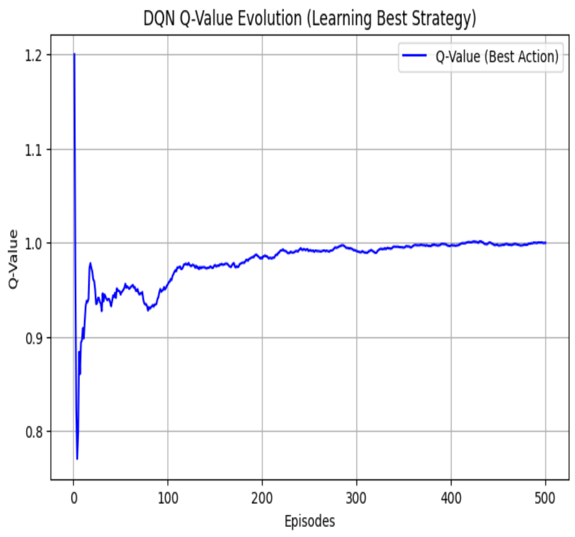


Figure 3: DQN training results at different episodes

Figure 3 presents the training performance of the DQN, which, through the neural network and Q-learning techniques, makes decisions on how to interact with the environment. Upon correct interaction, the model received a reward, while during wrong interaction, the model is penalised. From the results, it was observed that at several episodes, the model received rewards, which indicated successful and correct interaction with the environment. The learning strategy of the DQN was evaluated with the Q-value, measuring the actions of the model in learning interactions with the environment. Figure 4 presents the results. To measure the ability of the model to reduce attackers' exploits is monitored as shown in Figure 5.



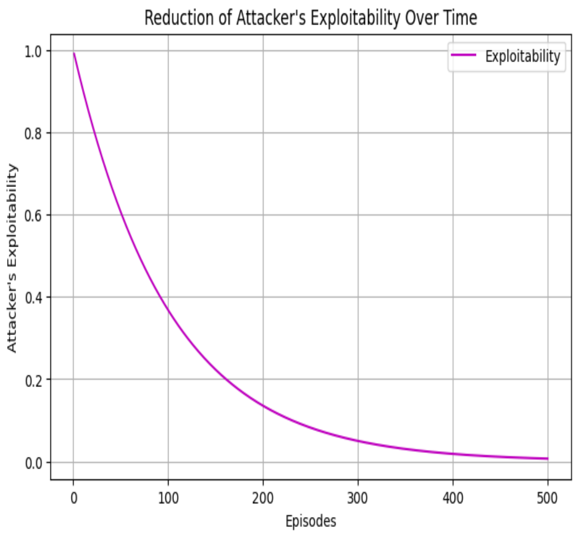


Figure 4: DQN learning evaluation results. Figure 5: Attackers' exploitability results

Figure 4 presents the DQN learning evaluation results at different episodes. From the results, it was observed that initially, the learning rate was unstable, which indicated that decision making was not consistent; however, after some time, the learning performance became stable, which is a good indication of the learning process. Secondly, the Q-values measure the action taken by the model during the learning process. From the results, it was observed that after 500 episodes, the Q-value was normalised to 1, which is very good and indicates a good learning process.

Figure 5 measured how the model was able to reduce the exploitation of attackers on the network. From the graphs, it was observed that while initially the exploit rate was very high, as the learning process continued, it was observed that the exploit rate was reduced to a normalised value which approximates zero. In Figure 6, the cumulative training rewards, action and detection success rates were reported.

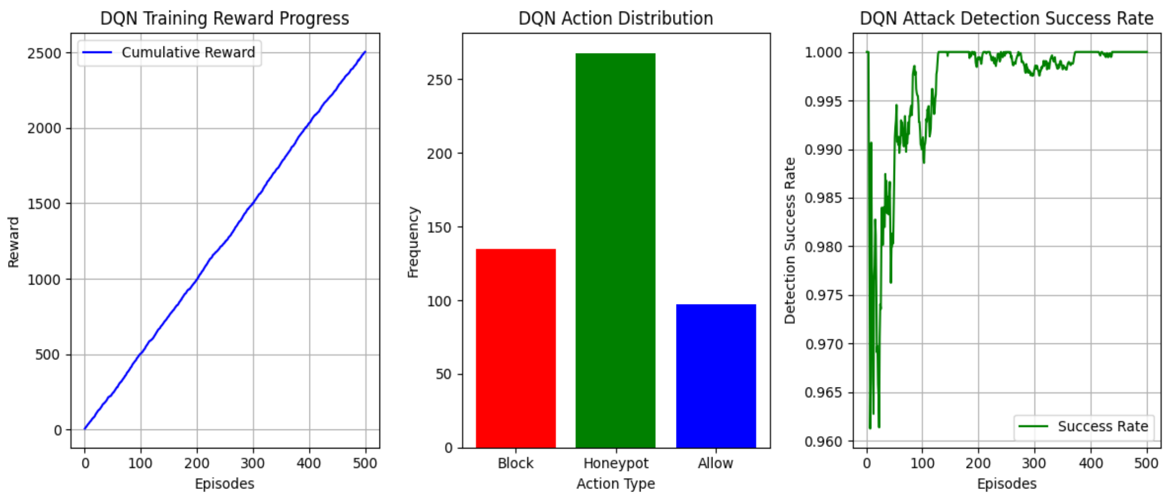


Figure 6: Training and detection result of the DQN model

The three sub-plots in Figure 6 present the training performance of the DQN, which records the cumulative rewards by the model during the training process. From the results, it was observed that at different episodes, the rewards recorded by the DQN were consistently increasing, indicating that as the training process continued, the model was able to improve its learning capabilities. The action of the DQN was also evaluated in the next results, showing frequencies at which packets are blocked or allowed by the honeypot, while the success rate of detecting attacks by the DQN was measured in the third graph on the left. From the results, it was observed that consistently, the model recorded an average of 0.975 accuracy in correctly interacting with the environment and classifying user actions.

2.1 Result of System Integration

The section presents the system integration when the adaptive honeypot was deployed on the online network facility for experimentation with different attack scenarios. Several attack scenarios were used to test the model. The attacker includes SQL injection, DDoS, brute force, XSS and reconnaissance threats. The attack detection rate was reported in Figure 7.

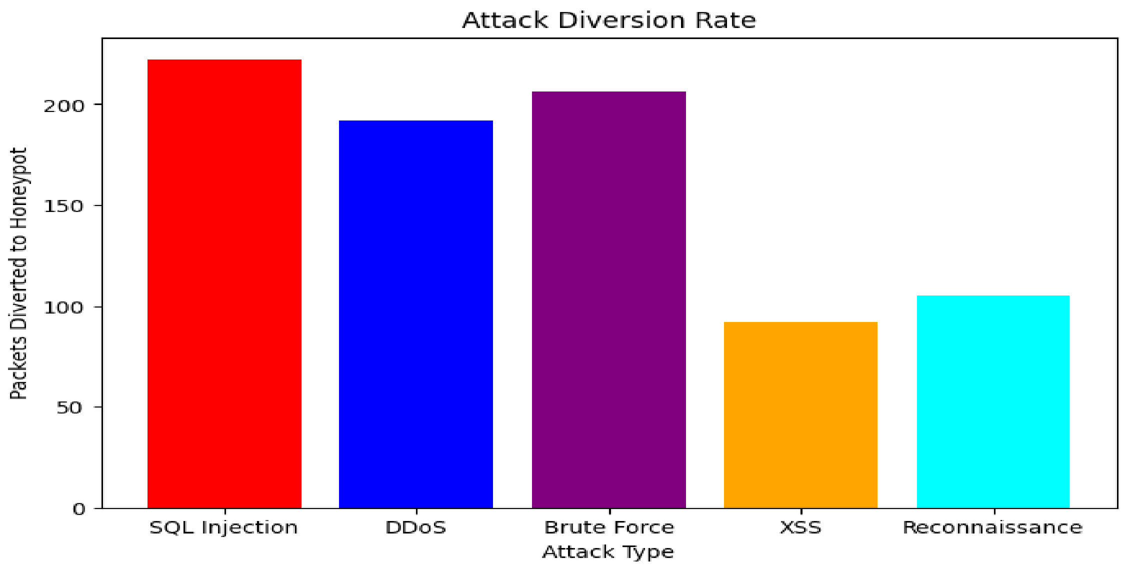


Figure 7: Experimental attack diversion rate

Figure 7 presents the attack detection rate. From the results, it was observed that several attack input injected into the network was identified and diverted to the honeypot. The reason was that the game theory was able to model the action of the attacker, while the DNQ interact to make appropriate and decisive action to make sure that the threats are detected and diverted to the honeypot. Figure 8 presents the honeypot Vs main server traffic flow, showing the number of packets diverted to the main server and those diverted to the honeypot. Figure 9 presents the honeypot strategy over time of training.

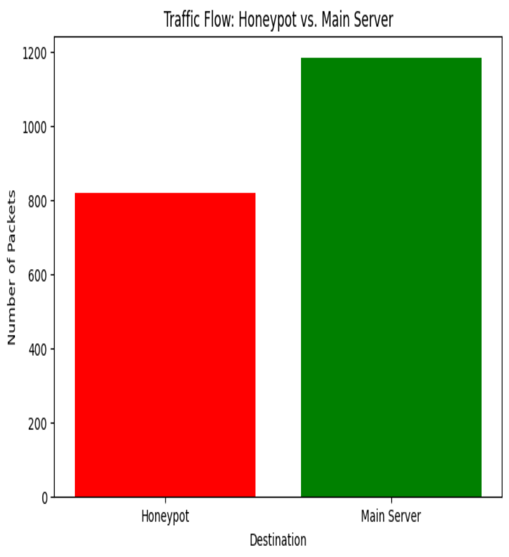


Figure 8: Honeypot vs main server

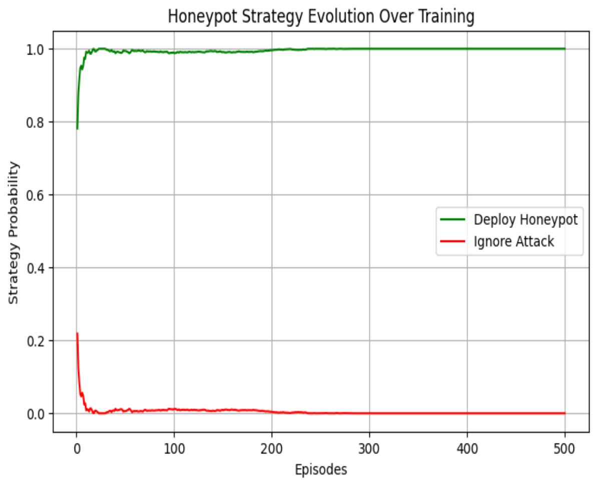


Figure 9: Honeypot interaction results during attack

Upon injection of a threat, it was observed that the packet classified as a threat was diverted to the honeypot environment, while those classified as legitimate packets were diverted to the main server. The graph showed how the honeypot behaves over time of attack injection on the network, and also attack features ignored due to false classification. From the result, it was observed that the honeypot deployment strategy was successful with a normalised value which approximates 1, and implied high threat detection success. In addition, the results also reported a very low and tolerable attack ignorance rate. This means that very low and tolerable attack was ignored, which makes this adaptive honeypot highly successful in mitigating threats.

III. CONCLUSION

This study successfully developed a reinforced honeypot technique for managing Advanced Persistent Threats (APTs) by integrating network modelling, game theory, and reinforcement learning into a dynamic and adaptive defence mechanism. The research demonstrated that traditional static honeypots can be significantly improved by incorporating intelligent decision-making and adaptive deception strategies. The implementation of a game-theoretic honeynet ensured that attacker interactions were strategically managed, prolonging engagement and enhancing intelligence collection. Furthermore, reinforcement learning enabled the honeypot to evolve, improving its effectiveness against sophisticated attack patterns. The integration of these models resulted in a robust system that not only diverted attackers but also gathered critical threat intelligence while continuously optimising its defence mechanisms.

Experimental results validated the efficiency of the proposed model. The system successfully engaged attackers for an extended duration, with an average session time of 7.2 minutes, outperforming conventional honeypots. Additionally, the threat intelligence analysis showed a 14% improvement in deception efficiency, ensuring better detection and analysis of malicious activities. The software implementation and testing confirmed the model's reliability, scalability, and resilience under high attack loads. Overall, the reinforced honeypot technique provides a promising approach to enhancing cybersecurity defences against APTs. By leveraging adaptive deception and strategic interaction models, this study contributes to the advancement of proactive security measures. Future work can explore further refinements, such as integrating additional machine learning techniques for more advanced attacker profiling and automating honeypot deployment in real-world enterprise environments.

REFERENCES

[1]. Amal, M., &Venkadesh, P. (2022). Review of cyber attack detection: Honeypot system. *Webology, 19*(1). https://doi.org/10.14704/WEB/V19I1/WEB19370

[2]. Ebere Uzoka Chidi, E Anoliefo, C Udanor, AT Chijindu, LO Nwobodo (2025)" A Blind navigation guide model for obstacle avoidance using distance vision estimation based YOLO-V8n; Journal of the Nigerian Society of Physical Sciences, 2292-229; https://doi.org/10.46481/jnsps.2025.2292

[3]. Ilg, N., Duplys, P., Sisejkovic, D., & Menth, M. (2023). A survey of contemporary open-source honeypots, frameworks, and tools. *Journal of Network and Computer Applications, 220*, 103737. https://doi.org/10.1016/j.jnca.2023.103737

[4]. Khoje, M. (2023). Securing data platforms: Strategic masking techniques for privacy and security for B2B enterprise data. *International Journal of Computer Trends and Technology, 71*(11), 46–54. https://doi.org/10.14445/22312803/IJCTT-V71I11P107

[5]. Lee, S. Y., & Park, J. M. (2024). Sampled-data stabilisation for networked control systems under deception attack and the transmission delay. *Communications in Nonlinear Science and Numerical Simulation, 131*, 107817. https://doi.org/10.1016/j.cnsns.2024.107817

[6]. Papaspirou, V., Maglaras, L., Ferrag, M., Kantzavelou, I., Janicke, H., &Douligeris, C. (2021). A novel two-factor honeytoken authentication mechanism. *arXiv*. https://doi.org/10.48550/arXiv.2012.08782v3

[7]. Sochima V.E., Asogwa T.C., Lois O.N., Onuigbo C.M., Frank E.O., Ozor G.O., Ebere U.C. (2025)" Comparing multi-control algorithms for complex nonlinear systems: An embedded programmable logic control application; DOI: http://doi.org/10.11591/ijpeds.v16.i1.pp212-224

[8]. Zhang, H., Gu, Z., Tan, H., Wang, H., Zhu, Z., Xie, Y., & Li, J. (2023). Masking and purifying inputs for blocking textual adversarial attacks. *Information Sciences, 648*, 119501. https://doi.org/10.1016/j.ins.2023.119501

[9]. Zuzcák, M., &Zenka, M. (2020). An expert system assessing the threat level of attacks on a hybrid SSH honeynet. *Computers & Security, 92*, 101784. https://doi.org/10.1016/j.cose.2020.101784