International Journal of Information Security (2026) 25:60
https://doi.org/10.1007/s10207-026-01221-5

REGULAR CONTRIBUTION

Q-Cowrie: An adaptive honeypot to analyse attackers' behaviour

Maryam Var Naseri[1] · Ian Welch[1] · Junaid Haseeb[2] · Masood Mansoori[3]

Received: 3 August 2025 / Accepted: 12 January 2026 / Published online: 20 February 2026
© The Author(s) 2026

Abstract

Honeypots in computer security have been used as effective security solutions to lure attackers, capture their interactions with the honeypot systems and study their behaviour. Attackers interacting with honeypots may use Artificial Intelligence (AI)-based techniques to detect the presence of honeypots leading to evasion by the attackers. This paper discusses the application of Reinforcement Learning (RL) to address these issues by improving response generation in honeypots. We propose "Q-Cowrie", a honeypot that is built upon customising a medium interaction server honeypot, that is, Cowrie, to increase the honeypot's deception. RL capabilities have been integrated into the honeypot to support adaptive behaviour while interacting with attackers. Two experimental studies have been conducted in which Cowrie and Q-Cowrie honeypots were used, respectively. First, we deployed a Cowrie honeypot to capture cyber attacks and identify attackers' goals and techniques. This allowed us to create a probabilistic model, that is, the Markov Decision Making Process (MDP), to understand the decision-making process of attackers in different situations. Learning from attackers' unique patterns and applying RL techniques, Q-Cowrie was able to actively interact with attackers, making adaptive decisions.

Keywords Reinforcement Learning · Cyber Attacks · Markov Decision Process (MDP) · Honeypots · Deception · Q-Cowrie · Cowrie

1 Introduction

Discovery of new vulnerabilities and rapid development of new exploits and attack techniques have increased the risk of cyber attacks [1]. Organisations around the world are confronting sophisticated and evolving threats that require the deployment of modern defence mechanisms [2]. Honeypots are defence mechanisms that provide valuable insights into attackers' behaviours, techniques, and methods and expand the understanding of threat landscapes [3]. Traditional honeypots maintain fixed-state configurations with no adaptability to attackers' dynamic behaviour. These static configurations are not ideal, as static responses to attackers' varying behaviour can expose the honeypot's presence [4]. Furthermore, static configurations must be frequently configured to respond to emerging attacker requests, and any failure in the update process can lead to detection by attackers [5]. Cyber adversaries also deploy Artificial Intelligence (AI)-based techniques to bypass security solutions and automate sophisticated attacks. Hence, static honeypots are inappropriate technologies for countering such advanced threats, as they cannot adapt or respond intelligently [4, 6].

Rapid advances in adversarial techniques have made static, rule-based honeypots ineffective due to their lack of adaptability required for meaningful interaction with attackers [6]. Despite the integration of AI and machine learning into honeypot systems, most existing adaptive honeypots continue to rely on predefined rules or static learning models, which constrain their ability to make autonomous and context-aware decisions. Furthermore, honeypots employing Q-Learning suffer from the cold start problem: with Q-tables initialised to zero or arbitrary values, these systems must learn effective policies through costly trial-and-error inter-

✉ Maryam Var Naseri
varnaseri.m@gmail.com

Ian Welch
ian.welch@vuw.ac.nz

Junaid Haseeb
junaid.haseeb@waikato.ac.nz

Masood Mansoori
m.mansoori@unsw.edu.au

[1] School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand

[2] School of Computing & Mathematical Sciences, University of Waikato, Hamilton, New Zealand

[3] Canberra School of Professional Studies, University of New South Wales (UNSW), Canberra, ACT, Australia

actions with real attackers, risking early detection before meaningful learning occurs.

These limitations motivate the current study to develop a self-learning honeypot capable of dynamically adapting to attacker behaviour while entering deployment with an informed initial policy. To address these gaps, this research proposes a self-learning honeypot Q-Cowrie built on a popular SSH/Telnet honeypot Cowrie[1] that integrates reinforcement learning (Q-Learning) with probabilistic modelling using Markov Decision Processes (MDP). The MDP, constructed from empirical attack data, enables offline simulation to derive informed initial Q-values before live deployment, combining model-based structure with model-free adaptivity. This approach enables the honeypot to enhance deception effectiveness, maintain longer engagement with attackers, and improve intelligence gathering in support of evolving cyber defence goals.

- **MDP-based adaptive honeypot framework:** We develop Q-Cowrie, an adaptive honeypot that integrates Q-Learning with a Markov Decision Process (MDP) model derived from empirical attack data. The MDP captures attacker transition probabilities across eight attack lifecycle states mapped to the MITRE ATT&CK framework, enabling standardised threat intelligence alignment.
- **Cold start mitigation through offline simulation:** We address the cold start problem inherent to Q-Learning honeypots by using the empirically-derived MDP to simulate realistic attack sequences offline, generating informed initial Q-values before live deployment. This enables Q-Cowrie to produce effective responses from the outset rather than learning through costly trial-and-error with real attackers.
- **Context-aware dynamic response generation:** We implement an autonomous response mechanism that selects contextually appropriate actions (allow, block, delay, insult) based on learned policy, generating realistic system feedback such as simulated DNS failures for blocked requests rather than obvious rejections.
- **Two-phase experimental validation:** We conduct a comparative experimental study with Phase 1 deploying standard Cowrie to collect baseline data and construct the MDP model, and Phase 2 deploying Q-Cowrie to evaluate adaptive response capabilities. Results demonstrate 94% successful dynamic response generation and 80% improvement in adaptability and learning compared to the baseline.
- **Attack pattern analysis for cryptomining and botnet threats:** We analyse over 400,000 attack sessions to identify unique attack patterns, demonstrating Q-Cowrie's improved capability to detect sophisticated download and execution attempts and classify complex attack sequences.

[1] SSH/Telnet Honeypot (https://github.com/cowrie/cowrie)

2 Background

This section provides the required background information for the research.

2.1 Honeypots

Honeypots are security solutions designed to simulate legitimate systems allowing attackers to interact with them and provide useful information to study their behaviour, tactics and techniques [7, 8]. *Conventional/Static honeypots* are configured and managed statically, requiring changes to be made manually [6, 9]. In contrast, *Dynamic honeypots* address the manual configuration and deployment shortcomings by adapting to the network changes and predefined attack patterns they capture [9]. However, their adaptability is limited due to the lack of autonomous decision-making against evolving threats [10]. To address this, *Self-adaptive honeypots* were introduced, enabling learning from attackers' behaviour and progress beyond static rule-based responses [11]. Although self-configuring honeypots simplify the deployment phase and adjust to dynamic changes automatically, their reliance on static rules limits their ability to combat evolving attacker strategies[12].

Honeypots are also classified according to their interaction level. *Low-interaction* honeypots replicate limited network services with restricted capabilities [9]. *Medium-interaction* honeypots are designed to involve more attackers compared to the low-interaction type [3]. *High-interaction* honeypots are fully functional systems that allow attackers to conduct malicious activities such as file uploads, malware installation, and command execution [13].

Cowrie is a widely used medium-high interaction honeypot designed to capture Secure Shell (SSH) and Telnet attacks by simulating a Debian-based installation. This allows attackers various functionalities including login interface, manipulate file system and execute commands through a command-line interface. As a result, attackers are led to believe the system is legitimate and initiate destructive activities such as downloading, installing, and modifying files [14]. While Cowrie allows for engaging attackers for longer periods [15], maintaining the engagement of the attacker is challenging as inconsistencies in the system's response can reveal their true nature [7]. To address this, our research applies RL MDPs to enhance deception and engagement.

2.2 Reinforcement Learning (RL) and Markov Decision Process (MDP)

Reinforcement Learning is a machine learning approach to learn from the environment for optimal decision making through trial and error by maximising cumulative rewards. In the environment, rewards serve as feedback, leading agents towards improved actions [16, 17]. According to [18], applying RL can lead to challenges in deceiving attackers due to the network uncertainties and lack of resources to learn from the environment. Therefore, applying MDP can address these limitations considering it is a sequential approach to model decision making [19, 20]. MDPs structure problems and define states, actions, transitions, receive rewards, and allow the agent to learn a policy that leads to adaptive behaviour [16, 17]. The agent uses a policy to select actions, while the value function estimates the future rewards [16]. Using MDP, the honeypot can dynamically adjust its responses based on past interactions, increasing the likelihood of engaging the attackers and delaying the honeypot detection. This adaptive approach improves honeypot effectiveness and allows richer behavioural data collection [21].

3 Related Work

Extensive research has explored types and applications of honeypots [22], and a growing body of work has examined reinforcement learning (RL) to strengthen the functionality of the honeypot, understand attacker tactics, profile adversaries and respond to evolving threats [23]. Although these studies have made substantial contributions to adaptive honeypot development, persistent gaps remain in areas such as autonomous decision-making, MDP-based policy initialisation, computational scalability, threat intelligence interoperability, and response realism. This section reviews previous work on adaptive honeypots, including early adaptive systems, reinforcement learning approaches, and MDP-based frameworks, before identifying key research gaps that motivate the current study.

3.1 Early Adaptive Honeypot Systems

Mohammadzadeh et al. [10] analysed the adaptability of honeypots and evaluated attackers' fingerprinting techniques, establishing the importance of avoiding fingerprinting through dynamic honeypot behaviour. The Heliza honeypot [24] was introduced to dynamically interact with attackers and profile them according to their identity and linguistic traits. Heliza used Markov chain modelling to generate contextual responses and incorporated provocative messages to encourage extended attacker engagement. However, Heliza demonstrated poor adaptability to unpredictable attacker behaviour and showed no capacity for autonomous decision-making, affecting the system's reliability. Furthermore, Heliza responded directly to attackers' input without learnt policy optimisation, exposing the system to potential vulnerabilities from erroneous output. The system implemented a basic learning algorithm that limited its ability to handle sophisticated attack patterns.

3.2 Reinforcement Learning Approaches

HARM [18] proposed an adaptive framework for autonomous response and detection against automated threats, deploying advanced RL algorithms including SARSA and Q-Learning agents integrated with honeypots to analyse malware behaviour. The authors identified honeypot detection by malware as a key challenge, noting that the interaction time is reduced quickly when attackers discover the honeypot, affecting the quality of collected data. The system also encountered redundant attack patterns when targeting automated and repetitive malware, requiring optimised filtering to maintain data integrity. Broken attack chains led to low-quality behavioural data, highlighting the need for more robust engagement mechanisms.

QRASSH [11] uses an adaptive strategy focused on human attackers with a significant improvement in detection and response. The system used deep Q-learning (DQN) for complete automation of the decision-making process, using an environment with 57 states representing Linux commands and 5 action states. However, QRASSH showed how complex and poorly defined reward functions in deep Q-Learning affect policy learning and adaptability, while data quality remained a key factor influencing the Q-Learning agent's effectiveness.

Dowling et al. [25] applied the SARSA RL algorithm to improve the honeypot interaction, particularly with botnet attackers. A reward-based mechanism was implemented to extend engagement and evade honeypot detection by emulating realistic and adaptive attacker behaviour. The system classified attackers' commands into bash and compound types, responding by allowing, blocking, or substituting commands accordingly. Although effective for basic engagement extension, the approach relied on categorical command classification rather than learnt contextual responses.

Navarro Ferrer [26] integrated RL algorithms such as SARSA and Q-Learning with deep learning techniques into Cowrie to develop a self-adaptive honeypot for threat intelligence gathering. Customised reward functions and data analytics significantly improved threat intelligence collection and maintained longer interaction with attackers. However, the system's adaptability remained constrained by the scope of predefined reward structures.

Suratkar et al. [34] presented an adaptive honeypot built on Cowrie using Q-Learning to increase attacker engage-

ment. The system categorised commands into six groups (CheckConf, Install, Download, Run, Password, ChangeConf) with actions of allow, block, delay, substitute, and insult. A severity-based reward function assigned fixed values: 1.0 for high-risk commands, 0.5 for medium-risk, and 0.0 for reconnaissance. The system demonstrated fast convergence toward optimal Q-values, requiring approximately 25 attacks versus 65 in comparable systems. However, limitations included the reliance on predetermined command categories, the absence of formal probabilistic modelling or threat framework mapping, and the focus on action selection without considering how the presentation of the response affects the perception of the attacker.

A common limitation of these reinforcement learning approaches is the cold start problem in Q-Learning and related algorithms. With Q-tables initialised to zero or arbitrary values, the agent lacks prior knowledge of which actions are beneficial in which states. Early on, action selection is therefore driven mainly by exploration rather than an informed policy, often yielding inappropriate, unrealistic, or easily detectable artificial responses. In honeypot settings, this is especially problematic, as poor early responses can reveal the system's deceptive nature before useful learning occurs. Even systems with faster convergence, such as Suratkar et al. [34], which report convergence within 25 attacks for some actions, still depend on real attack interactions to learn effective policies. None of the systems reviewed explicitly consider how initial policy quality affects early deployment or propose methods for informing Q-value initialisation before live operation.

3.3 MDP-Based Approaches

Hayatle et al. [27] proposed an MDP model for the strategic deployment of high-performance honeypots to manage botmasters. The model aimed to balance threat intelligence collection with minimisation of legal issues caused by illicit acts within the honeypot. The MDP framework enabled decisions on state transitions to accept or reject botmaster commands. To handle uncertainty in honeypot states, they deployed a Partially Observable Markov Decision Process (POMDP) model, allowing strategic decisions to be made with partial knowledge. Their results demonstrated that the model could extend the interaction with botnet attackers while reducing detection and legal risks.

Pashaei et al. [28] used an MDP framework to develop a SARSA-based Deep Reinforcement Learning model to improve honeypot-based intrusion detection in Industrial Control Systems. The model targeted DDoS and Man-in-the-Middle attacks through classification and environmental agents to improve learning. MDP-based interaction modelling allowed for optimisation of decision-making while capturing and gaining a better understanding of attackers' behaviour, achieving improved precision and F-measure scores compared to traditional intrusion detection systems.

MDP-based honeypot approaches highlight the value of probabilistic modelling but do not solve Q-Learning's cold start problem, and prior work has not used MDPs to set initial Q-values. This study builds an MDP from empirical attack data to capture transition probabilities that reflect real attacker behaviour, then runs offline simulations to derive informed initial Q-values before deployment. This combines a model-based structure with model-free Q-Learning's adaptivity, while previous work treats MDP formulation and Q-Learning initialisation as separate tasks.

3.4 Summary

A review of prior studies identifies four principal gaps in the existing research.

- **Fixed rules and reward schemes:** Existing approaches typically rely on predefined command categories and static reward structures, which constrain their ability to adapt autonomously and contextually to previously unseen attacks.
- **Absence of rigorous probabilistic modelling:** Attacker behaviour is rarely captured using explicit MDP formulations grounded in empirical data, undermining both the theoretical foundations and the systematic study of attacker decision processes.
- **Cold-start limitations and weak initial policies:** Honeypots based on Q-learning usually begin with arbitrary Q-values and learn exclusively from real-time attacker interactions, leading to inefficient behaviour in early stages and overlooking the potential of offline, MDP-driven simulations to initialise policies more effectively.
- **Lack of standards:** Use of idiosyncratic state representations hinder mapping honeypot observations to standard schemas limit integration with existing security ecosystems and reducing the value of the resulting threat intelligence.

4 Experimental Design

4.1 Overview

In this research, we conducted two experimental studies to identify the baseline and create an MDP model to integrate into the honeypot and enable learning from interactions with attackers through the application of RL algorithms (Q-Learning algorithm) and subsequently responding to attackers' actions.

We conducted a two-phase experimental study to validate our adaptive honeypot approach. Phase 1 deployed standard Cowrie honeypots to collect baseline attack data, while Phase 2 integrated our Q-Learning-based MDP model into Cowrie (termed "Q-Cowrie") to evaluate adaptive response capabilities.

4.2 Evaluation Metrics

We do not use traditional classification metrics (precision, recall, F1-score) because they are fundamentally misaligned with RL evaluation. Classification metrics assume independent samples with binary outcomes, but Q-Cowrie optimises cumulative rewards over multi-step episodes with complex temporal dependencies across an 8-state model. Our goal is learning optimal response policies that maximise long-term engagement while preventing compromise–measured through reward convergence and policy stability rather than classifying individual attacker actions as correct or incorrect.

Instead, we evaluate Q-Cowrie through cumulative reward trends measuring learning effectiveness, policy convergence assessing response strategy stability, state transition accuracy validating the MDP model against real attacker behaviours, session engagement metrics including command diversity and interaction duration, and security outcomes such as resource hijacking prevention rates. This approach aligns with standard RL evaluation practices and better captures adaptive honeypot performance in dynamic threat environments than static classification metrics designed for supervised learning tasks.

4.3 Enhancements in Deception Levels

The level of deception was increased over a standard honeypot in two distinct phases. The first phase involved strategic configuration of a standard Cowrie honeypot, while the second introduced an adaptive, intelligent agent named Q-Cowrie.

In the initial experiment, a standard Cowrie honeypot was modified to be more attractive and believable to attackers, specifically those engaged in cryptomining and botnet activities. The configuration was optimised to attract attackers by simulating a vulnerable and easily exploitable system through exposed weak credentials with common username-password combinations, simulated vulnerable services via open network ports, and realistic system emulation where the SSH banner mimicked a Debian 11 system with a populated file system resembling a legitimate production server.

Beyond luring attackers, the deception level was increased by making interactions feel more authentic. The command-handling module was modified to introduce randomised delays of 3–5 seconds in command responses, mimicking natural system or human latency to make the honeypot feel less like an automated, instant-response system. Additionally, the system provided varied outputs to commands, avoiding the static and repetitive responses that can reveal a standard honeypot's nature.

Instead of relying on static, pre-programmed rules, Q-Cowrie can autonomously decide how to react to an attacker's actions to maximise engagement and intelligence gathering. The primary increase in deception comes from Q-Cowrie's ability to generate dynamic, context-aware responses. The system learns an optimal policy to decide whether to allow, block, delay, or insult an attacker based on their commands.

For non-malicious or reconnaissance commands (e.g., downloading a benign file), Q-Cowrie allows the response by simulating a successful action with realistic feedback (e.g., `Connecting to... connected`, `200 OK`), building the attacker's trust and reinforcing the perception of a real system. When a known malicious URL or action is detected, Q-Cowrie blocks the response by providing deceptive feedback that simulates a realistic network failure, such as a Domain Name Resolution (DNS) error (`wget: unable to resolve host address`), misleading the attacker into believing the issue is with their target URL or the network rather than the honeypot itself. To mimic network latency or a busy server, Q-Cowrie can delay responses to certain commands, enhancing realism and potentially lulling attackers into a false sense of security, believing their actions have gone undetected on a slow machine. Finally, inspired by the Heliza honeypot, an experimental insult feature was designed to frustrate attackers or probe for human interaction by responding to certain commands with taunts (e.g., for the `chmod` command: "Changing permissions won't change your chances"), adding a layer of unpredictable, human-like interaction.

4.4 Experiment 1

To study the techniques of the attackers, we deployed two instances of Cowrie honeypot, which is a medium-interaction SSH and Telnet honeypot [14, 29, 30]. The honeypots were operational for a duration of 21 days, and their configuration was optimised to capture cryptomining and Botnet attacks based on a review of these types of attacks (e.g. high processing power for cryptomining and vulnerable versions of the operating system). Weak credentials and commonly targeted ports were exposed to simulate vulnerable services and easily exploitable accounts. Additionally, verbose logging was enabled to capture and analyse attacker behaviours in detail for an in-depth later analysis. Actions related to downloading, executing, storing, and modifying malicious files (e.g., miner, locker) on the honeypot were observed in the captured log files. The data collected from this experiment helped to create the MDP model.

4.4.1 MDP Model Formulation

Various components are involved in designing the MDP model, such as states, actions, transition probabilities and rewards.

- **State** (s): A variable representing the attacker's objective (e.g., Initial Access, Execution, Persistence) [31].
- **Action** (a): A variable representing tactical steps that attackers use to transition between states. These actions map to MITRE ATT&CK framework techniques, such as exploiting public-facing applications or executing malicious scripts [31].
- **Reward function** (R): A function that assigns numerical values to pairs of state actions, where $R(s, a)$ quantifies the immediate value of taking action a in state s [31, 32].
- **Transition probabilities** (P): A function describing state transition likelihoods, denoted as $P(s \rightarrow s')$, where s and s' represent current and next states, respectively [31]. For example, using a "command and script interpreter" technique transitions from "execution" to "Credential Access" state with probability $P = 0.7$ (70%).
- **Discount factor** ($\gamma = 0.9$): A parameter bounded between [0,1] that determines how future rewards influence current decisions [31]. A high value $\gamma (\approx 1$ supports progressive decision-making, which is a fundamental factor in dynamic attacker–defender environments where delayed rewards capture the effectiveness of ongoing deception. Similar parameters have been applied in RL-based security studies, including network defence and adaptive honeypots, to formulate continuous interactions with attackers [33, 34].
- **Learning rate** ($\alpha = 0.1$): A parameter that controls the rate of model update during training, balancing the weight between new and existing information [31]. Therefore, a moderate α to balance stability and adaptability, while allowing adaptive updates. Similar small learning rates are widely used in reinforcement learning for security applications and intrusion detection to achieve reliable learning in uncertain environments. [33, 35].
- **Epsilon** ($\epsilon = 0.1$): A parameter that balances exploration and exploitation in reinforcement learning [36]. Setting $\epsilon = 0.1$ allocates 10% of actions to random exploration and 90% to exploiting known optimal actions [31]. This setting suggests the agent explores new actions 10% of the time and uses known optimal actions 90% of the time. This balance supports stable learning and prevents early convergence, and aligns with prior RL applications in cyber defence and network security [33, 35].
- **Episodes** ($N = 1000$): A parameter that defines a complete sequence of states, actions, and rewards from time $t = 0$ to T [36, 37]. This value optimises the trade-off between training time and learning performance.
- **Maximum steps** ($M = 100$): A parameter that specifies the maximum number of actions allowed within a single episode [31, 36]

The actions we are considering in MDP models have been mapped to MITRE ATT&CK Framework, Before we discuss the integration of MDP in honeypot for the second experiment, we will look into the attack actions mapping.

4.4.2 Mathematical Modelling for MDP Model

We represent the MDP formulation into a mathematical model representing the different components of the system as follows:

- **State space:**

  $$S = \{S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7\},$$

  where each S_i corresponds to the MITRE stage listed in Table 1.
- **Action sets:**

  $$A(S_i) = \{A_{i+1}\}, \quad i = 0, 1, \ldots, 6$$

  $$A(S_7) = \{A_{\text{stay}}\}$$
- **Transition probabilities:** For each observed transition $(s, a) \rightarrow s'$, the transition probability is defined as

  $$P(s' \mid s, a) = \Pr(S_{t+1} = s' \mid S_t = s, A_t = a).$$

  We estimate these probabilities empirically as

  $$\hat{P}(s' \mid s, a) = \frac{\left|\{t : S_t = s,\ A_t = a,\ S_{t+1} = s'\}\right|}{\left|\{t : S_t = s,\ A_t = a\}\right|},$$

  representing the observed frequency of $(s, a) \rightarrow s'$ in the dataset.
- **Reward function:** Each transition $(s, a) \rightarrow s'$ receives a reward $R(s, a, s')$ based on tactical severity, as summarised in Table 3:

  $$R(s, a, s') = \begin{cases} +5, & \text{High severity,} \\ +2, & \text{Medium severity,} \\ +1, & \text{Low severity,} \\ 0, & \text{Otherwise.} \end{cases}$$

Table 1 Attack States Used in MDP Model

State	State Name	Description
S0	Initial Reconnaissance	The attacker compiles information about the system, identifying open ports, services, and potential vulnerabilities.
S1	Initial Compromise	The attacker breaches the system through exploitation or credential abuse to gain initial access.
S2	Establish Foothold	The attacker maintains a presence on the host, often by installing backdoors or creating new user accounts.
S3	Privilege Escalation	The attacker gains higher-level privileges to expand control over the compromised environment.
S4	Defence Evasion	The attacker attempts to evade detection mechanisms such as logging, monitoring, or antivirus tools.
S5	Resource Hijacking	The attacker diverts system resources (e.g., CPU/GPU) for cryptomining or other unauthorised computations.
S6	Command and Control	The attacker establishes communication with a remote server to issue commands and manage the mining or malicious activities.
S7	Persistence	The attacker deploys techniques (e.g., startup scripts, hidden cron jobs) to maintain long-term access to the system.

We set the discount factor $\gamma = 1$, so that the total return of any path is the sum of its transition rewards.

4.4.3 MITRE ATT&CK Framework Mapping

The MITRE ATT&CK framework [38] is a worldwide repository of adversary techniques and tactics. To extend the parametrising of the MDP model, each state in our MDP corresponds to a specific phase, and each action corresponds to a tactic in the framework.

States Reflecting Attack Lifecycle Phases The MITRE ATT&CK framework includes states such as Initial Access, Execution, Persistence, Privilege Escalation, Defence Evasion, Credential Access, Discovery, Lateral Movement, Collection, Command and Control, Exfiltration, and Impact [38]. We identified the states corresponding to specific techniques that are most relevant to the attack scenarios and fit within cryptomining and botnet attacks. Table 1 lists the relevant states for the scope of this research.

In our approach, we chose eight states (S0 - S7) from the MITRE ATT&CK framework, which include the tactics involved in cryptomining and bot attacks. Each state represents a critical phase in the attack lifecycle, enabling us to detect and respond to malicious activities systematically. The correlation of S0-S8 with phases in MITRE ATT&CK framework ensures that our honeypot can effectively monitor and log events corresponding to these phases.

Actions Reflecting Tactics in the Attack Lifecycle Phases We extracted commands that the cryptomining and botnet attackers executed to map them with actions. Figure 1 illustrates the workflow of extracted states and actions in our data set's Cryptomining and Botnet attack scenarios. We analysed and refined nearly 249,000 sessions by applying a multistep approach involving data preprocessing, pattern recognition, and attack pattern classification. We deployed Python-based log parsers to identify patterns, measure the frequency of attack techniques, evaluate source IP reputation, and analyse command execution patterns. In addition, for the refinement, we eliminated redundant data for deeper analysis and grouping and labelling sessions aligned with attack stages based on the MITRE ATT&CK framework, forming a solidly structured data set for further threat analysis. The list of relevant actions for the scope of this research can be found in Table 2.

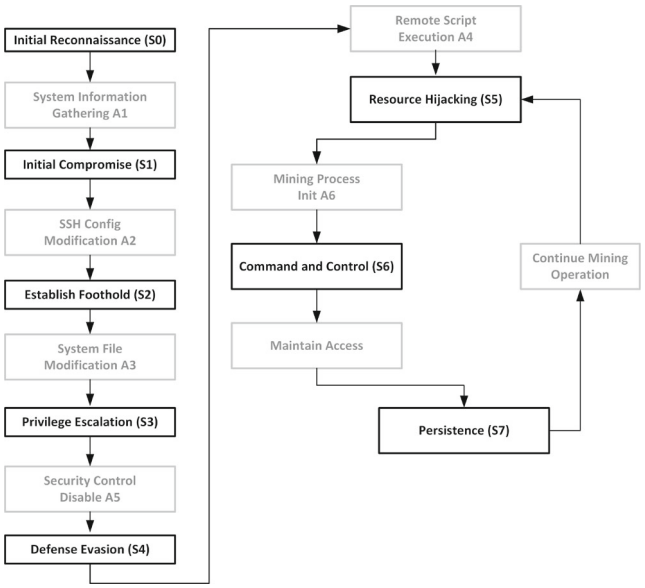

Fig. 1 Workflow diagram for states and actions in cryptomining and botnet attacks

Rewards Allocation In the proposed MDP-based honeypot model, the reward function evaluates the desirability of state–action outcomes according to observed attacker behaviour. The reward structure is designed to align with the command patterns exhibited by adversaries, promoting defensive engagement in response to high-risk activities. Table 3 presents the defined reward levels and their corresponding assignment criteria.

Transition Probabilities Transition Probabilities (P) in MDP indicate the likelihood of transiting from one state to another through a specific action. Formula (1) represents the Transition Probabilities equation.

$$P(s' \mid s, a) = \Pr(St + 1 = s' \mid St = s, At = a) \quad (1)$$

We use this equation to calculate transition probabilities from state s to state s' by action a $P(s' \mid s, a)$ where St and At denote the state and action at time t accordingly. St+1 denotes the state at time t+1.

Mapping of States and Actions The connection between states and actions and the behaviour and techniques demonstrated by cryptomining and botnet attackers are mapped to the MITRE ATT&CK framework for the scope of this research in Table 4. The table represents the attackers' objectives at each stage, the applied tactics (actions) used to achieve those objectives, and the corresponding MITRE

Table 2 Attack Actions Defined in MDP Model

Action	Action Name	Description
A1	System Reconnaissance	Commands used to gather system information, e.g., `uname -a` and `grep "model name" /proc/cpuinfo`, to identify hardware, OS, and service details.
A2	SSH Configuration	Commands that modify SSH settings or use SSH to move laterally within the network (e.g., adding keys, altering `sshd_config`).
A3	Modify System Files	Access or modification of sensitive files such as `/etc/shadow` or `/etc/passwd` for privilege escalation or persistence.
A4	Execute Remote Scripts	Downloading and executing remote scripts (e.g., using `curl <URL> | sh`), potentially deploying cryptomining scripts or toolchains.
A5	Disable Security Measures	Commands that change file attributes or disable protections (e.g., `chattr -ia <file>`) to weaken detection and recovery.
A6	Resource Manipulation	Initiating or managing cryptomining processes or other resource-intensive tasks to hijack CPU/GPU cycles.

Table 3 Defined Rewards in MDP Model

Reward Degree	Description
High (+5)	Detection of downloads or executions of known malicious payloads, such as `curl [URL]` linked to cryptomining domains or exploit servers.
Medium (+2)	System reconnaissance activities including commands like `uname -a` or `grep "model name" /proc/cpuinfo`, indicating environment probing behaviour.
Low (+1)	File or permission modification operations (e.g., `chattr -ia .ssh`) that pose limited or delayed threat impact.

ATT&CK techniques (states) associated with those actions. Specifically, the State (Technique) describes various attack stages and the attackers' objectives, while the Action (Tactic) outlines the applied methods used to accomplish those objectives.

The pseudocode in Algorithm 1 illustrates the log parsing and cleaning process, where raw Cowrie honeypot logs are normalised, mapped to MITRE ATT&CK techniques, and transformed into a structured dataset. As depicted in the pseudocode, we initially parse and normalise the log files extracted from the honeypot.

Algorithm 1: Parsing and Cleaning Cowrie Honeypot Logs

```
BEGIN;
DeployCowrieHoneypot();
INPUT CowrieRawLogs;
DATASET cleanData;
for log ∈ CowrieRawLogs   // Parse all Cowrie logs
do
    cleanData ← ParseJSON(log);
    record.command ← ExtractCommand(cleanData);
    Normalise(record.command);
    technique ← MapToMITRE(record.command)      // Map
    command to MITRE Append(cleanData, [timestamp,
    session_id, src_ip, technique]);
CALL StoreDataset(cleanData);
END;
```

The pseudocode in Algorithm 2 calculates the transition probabilities between these states, and producing the state transition matrix required for MDP modelling.

4.5 Experiment 2

Data collection and analysis from experiment 1 facilitated the development of the MDP model, which was then integrated with the Cowrie honeypot to support the next study phase. In experiment 1, we studied cryptomining and botnet attackers and the findings allowed us to develop MDP using a RL algorithm (i.e. Q-Learning) integrated with Cowrie to form what we term Q-Cowrie.

Algorithm 2: Computing Transition Probabilities from Cleaned Cowrie Logs

```
BEGIN;
INPUT cleanData;
MATRIX transitionCount, transitionProb;
for session ∈ UniqueSessions(cleanData) do
    events ← GetSortedEvents(cleanData, session);
    for i = 1 to Length(events) − 1      // Loop through
    events do
        from ← events[i].technique          // Get current
        technique;
        to ← events[i+1].technique              // Get next
        technique;
        transitionCount[from][to] ← transitionCount[from][to] +
        1      // Increment transition count;
for from ∈ transitionCount  // Loop over each source
state do
    total ← Sum(transitionCount[from])          // Sum all
    transitions from this state;
    for to ∈ transitionCount[from]            // Loop over
    destination states do
        transitionProb[from][to] ← transitionCount[from][to] /
        total  // Compute normalised probability;
CALL StoreMatrix(transitionProb)          // Save the
transition probability matrix;
END;
```

4.5.1 Implementing Q-Learning for MDP modelling - Development of Q-Cowrie

Our MDP model aims to develop an autonomous approach to solving decision making issues in the dynamic threat landscape. The learning model is designed to formulate policies,

Table 4 Mapping of States and Actions to MITRE ATT&CK Techniques

State (Techniques)	Action (Tactics)	Description and MITRE ATT&CK Technique
Initial Reconnaissance (S0)	**System Reconnaissance (A1)**	The attacker gathers host and environment information such as OS type, CPU details, or active services. **Technique:** T1082 – *System Information Discovery*.
Initial Compromise (S1)	**SSH Configuration (A2)**	The attacker gains system access by exploiting external remote services or weak SSH configurations. **Technique:** T1133 – *External Remote Services*.
Establish Foothold (S2)	**Modify System Files (A3)**	The attacker maintains access by modifying startup scripts or system files to enable persistence. **Technique:** T1546 – *Event-Triggered Execution*.
Privilege Escalation (S3)	**Modify System Files (A3)**	The attacker escalates privileges using shell commands or scripting interfaces to gain administrative control. **Technique:** T1059 – *Command and Scripting Interpreter*.
Defense Evasion (S4)	**Disable Security Measures (A5)**	The attacker attempts to disable security mechanisms such as antivirus tools or system logs to remain undetected. **Technique:** T1562 – *Impair Defenses*.
Resource Hijacking (S5)	**Resource Manipulation (A6)**	The attacker misuses system resources (e.g., CPU/GPU) to perform cryptomining or botnet operations. **Technique:** T1496 – *Resource Hijacking*.
Command and Control (S6)	**Execute Remote Scripts (A4)**	The attacker downloads and executes external scripts to establish remote control channels. **Technique:** T1105 – *Ingress Tool Transfer*.
Persistence (S7)	**Modify System Files (A3)**	The attacker ensures continued access through account manipulation or modification of persistent configuration files. **Technique:** T1098 – *Account Manipulation*.

essentially a plan for choosing actions that extend long-term benefits. Such improvements will allow the model to more precisely emulate and adjust to the unpredictable threat actions and associated responses in the context of a Cowrie honeypot instance.

We created a q-algorithm (q-table) to design the learning agent and completed the training phase. Based on the final q-table, we extracted the optimal policy. The q-table is initialised with values "0" to be used in the Q-Learning algorithm, where these q values will be updated based on the rewards received from actions taken in different states. The values are initialised to 0 as part of the Q-Learning process, which will be updated based on the rewards obtained during the training phase. Figure 2 shows the visual representation of the initialisation of the q-table as described.

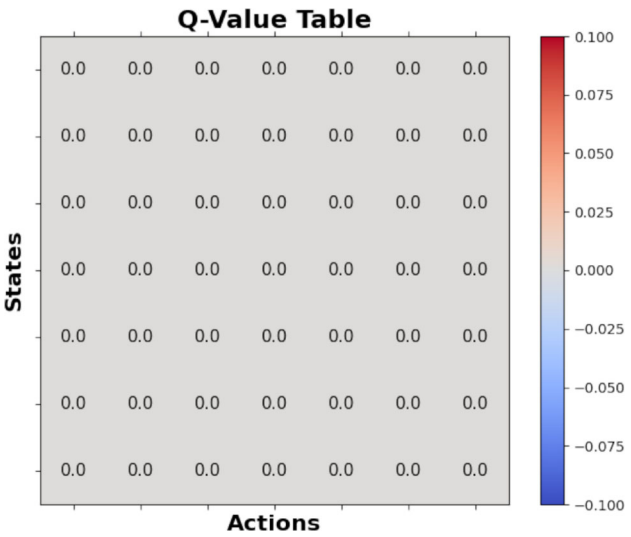

Fig. 2 Q-Table initialisation and optimal policy extraction

5 Experimental Deployment

The primary goal of our experiments was to conduct an evaluation study on the modified Cowrie honeypot (experiment 1) with our MDP model developed using the Q-Learning RL algorithm (experiment 2). The evaluation focused on the effectiveness of attacker detection and the generation of appropriate responses to attackers. In the following, we outline the methodology and processes involved in two distinct experiments.

Experiment 1 The objective of the first experiment was to modify the configurations of the honeypots to attract crypto mining and botnet attackers. This experiment aimed to create

an environment that simulates real-world vulnerabilities and collects rich datasets for further analysis.

Experiment 2 This was designed to simulate and analyse targeted attack scenarios, including crypto mining and Botnet. The data collected was used to evaluate the Q-Cowrie model, which integrates the Cowrie honeypot with an MDP-based model developed using a Q-Learning algorithm. The objective was to test the system's ability to dynamically adapt and generate responses while interacting with cryptomining and botnet attackers based on the learnt policy.

We set up a Cowrie honeypot for cryptomining and bot execution attack scenarios. The decision was based on observation derived from our first experimental study as it suggests the attackers' patterns were aligned with cryptomining and botnet behaviour. By selecting these scenarios, our goal was to address two prevalent and high-impact threats in the context of cybersecurity. 1) Cryptomining attackers exploit the system resources for unauthorised mining activities, resulting in financial and operational disruptions. Therefore, through this study, our aim was to identify exploited vulnerabilities and develop security measures to mitigate resource consumption and financial loss. 2) Similarly, botnets are significant threats as they can be used as a platform to launch DDoS attacks, data theft, etc. [40]. Therefore, deploying the Cowrie honeypot can help to understand the behaviour of the bot for an in-depth analysis of their attack vector, Command and Control (C&C) protocols, and propagation techniques.

5.1 Experiment 1 - Cowrie Honeypot with Modified Configuration

In our first experiment, the Cowrie honeypot was deployed with modified configurations on a physical host machine that was openly accessible by attackers over the Internet. The Cowrie deployed to log attackers' activities with Cowrie's configuration file, aiming to give the attackers fake information when they gather information about the system, OS, user, open ports, services, etc. In this experiment, the SSH banner of cowrie was set to mimic a Debian 11 system. The file system included realistic Linux files and directories. Additionally, the command-handling module (cowrie.shell.commands) was modified to introduce human-like randomised delays (e.g., 3–5 second response times using Python's time.sleep) and varied outputs to attackers' commands.

5.1.1 Data Collection

Our compiled data set before integrating MDP includes an approximate number of 410,198 attack sessions that had been logged for nearly 3 weeks in time of data collection. The data set was prepared in JSON format, providing information such as timestamp, ports targeted, login credentials of honeypot and attempted commands for all attacks.

5.2 Experiment 2 - Q-Cowrie Honeypot

In our second experiment, the Q-Cowrie (our Q-Learning based Cowrie) honeypot was deployed on a physical host. The honeypot was deployed to be openly accessible by attackers over the Internet. Our Q-Cowrie is deployed to study attackers' actions and manage responses based on the monitored states and actions. In this experiment, we evaluated and justified the selection of RL parameters for the Q-Cowrie system for reliable and effective policy learning. The learning rate (α) was set to 0.01, which offered a balance between convergence speed and stability, preventing oscillations in the learned policy. The discount factor (γ) was set to 0.95 to prioritise long-term outcomes of deception actions, which is crucial in scenarios where future attacker states strongly influence honeypot effectiveness.

The exploration rate (ε) was initially set to 0.1 to achieve an appropriate exploration–exploitation trade-off. Through preliminary tests over 100 episodes, we observed that this value produced stable cumulative rewards and consistent policy convergence. Informal tuning experiments varying ε between 0.05 and 0.3 confirmed that $\varepsilon = 0.1$ achieved the most reliable attacker engagement and realistic honeypot behaviour. These findings validate our chosen hyperparameters as suitable for dynamic deception in real-world honeypot operations.

Furthermore, in experiment 2, the Cowrie honeypot provides responses to allow attackers' requests, block, or delay to provide information to the attackers. It can also send insult messages to attackers as a response mechanism to encourage them to try alternative ways to get information. In this research, we implemented custom insult messages triggered by specific commands. For example, when attackers attempt to use commands like `chmod`, `top`, `chattr`, `free -m`, or `sudo rm -rf`, the system responds with targeted messages:

- For `chmod`: "Changing permissions won't change your chances"
- For file modifications: "Trying to modify files? Too bad it's not allowed"
- For deletion attempts: "Delete away; it's not like you'll succeed"
- For `top`: "Monitoring processes? Here's one: 'YOU' failed"
- For resource queries: "System resources? Better save some dignity instead"
- For general attempts: "Nice try, but that's not going to work"

Moreover, Q-Cowrie contains custom scripts to mimic vulnerable services, a configuration to allow unlimited inbound and outbound traffic, and the ability to log all interactions with the Q-Cowrie honeypot. These include a full simulation of the SSH and Telnet protocols, including the authentication and logging capabilities of the credentials in the authentication process, the logging commands issued by the attacker, the generated response generated by the honeypot, and the duration of each attack. It also supports many Linux commands, allowing an attacker to perform reconnaissance on the honeypot system, download, upload, and unzip files, create files and directories, and perform a wide range of other system and network operations.

5.2.1 Data Collection

Our Q-Cowrie dataset includes 248,338 attack sessions for analysis within three weeks for cryptomining and Botnet data. The data set provided us with a sample of malicious patterns and behaviour. The data set was prepared in JSON format, providing information such as timestamp, targeted ports, login credentials of honeypot, and attempted commands. Our observations show that most of the attack patterns fit into the botnet and crypto mining scenarios. Therefore, we consequently shifted our research focus to these scenarios.

6 Data Analysis and Evaluation Metrics

Data collection and quantitative results from experiment 2 allowed for a comprehensive analysis of evaluation metrics. It enabled us to gain insight into performance evaluation of pre and post integration in attack pattern trends, frequency of the executed commands, learning stability, and responses based on observed actions.

6.1 Core Metrics and Evaluation Criteria

The number of core metrics and criteria facilitates evaluating the effectiveness of the post-integration system (aka Q-Cowrie). We follow scenario-based evaluation, statistical analysis, and comparative assessment. Our experiment targeted Cryptomining and Botnet attackers and studied Cowrie's performance before and after the integration. Using the data collected from the second experiment, we will evaluate Q-Cowrie's ability to detect, adapt, learn, and respond to attackers.

Quantitative metrics such as session attack counts, frequency of executed commands, and their techniques are incorporated to identify unique patterns and determine attackers' adaptability when interacting with Q-Cowrie. Furthermore, numerical values, including Q values and reward distributions, are measurement criteria to assess the effectiveness of the RL model. Our evaluation framework leverages log-based analysis to identify new pattern trends and the system's response behaviour.

Identify Unique Patterns The Q-Cowrie's ability to detect new or modified attack patterns over time. Q-Cowrie establishes the learning from previous interaction patterns for future pattern recognition.

Adaptability and Learning The Q-Cowrie's ability to establish high adaptability by dynamically manipulating its system information and CPU details to meet the specifics the attacker expects to get. For example, if an attacker executes the "lscpu" command, the honeypot should provide the configuration of common vulnerable settings the attacker expects to exploit.

Response generation The decision made by Q-Cowrie to respond to the attacker. For instance, if the attacker intends to download a malicious file or script from a suspicious URL through 'wget' or 'curl' commands, Q-Cowrie should decide to allow, block, delay, or terminate this interaction.

The algorithm 3 summarises the end-to-end process for data analysis, MDP training, and metric-based evaluation used in our experiments. The workflow enables comparison of key evaluation metrics, such as adaptability and response, between the offline baseline and the live online model.

Algorithm 3: MDP Training, Deployment, and Evaluation Process

BEGIN;
INPUT $Dataset\_A$, $Dataset\_B$, Metrics;
for $data \in \{Dataset\_A, Dataset\_B\}$ **do**
  CleanAndMap(data);
// Offline MDP Training
$Q_A, P_A \leftarrow TrainMDP(Dataset\_A)$ // Train MDP on Dataset A;
EvaluateConvergence(Q_A);
Store(Baseline.Q_values = Q_A, Baseline.Transitions = P_A);
// Honeypot Deployment and Online Update
Dataset_B ← DeployHoneypotAndCollect();
$Q_B, P_B \leftarrow UpdateandLearnMDP(Dataset\_B)$;
// Comparing Metrics Between Offline and Online Models
for *each Metric* $\in$ *Metrics* **do**
  Offline ← Compute(Metric, Q_A, P_A) // Offline metric;
  Online ← Compute(Metric, Q_B, P_B) // Online metric;
  Display(Metric + ": " + Offline + " vs " + Online);
Compare(Q_A, Q_B, P_A, P_B);
PlotResults();
END;

6.2 Quantitative Results

This quantitative analysis evaluates the effectiveness of integrating the MDP model with the Cowrie honeypot. Therefore, we will provide an analysis and discussion of the evaluation metrics to demonstrate the enhancement after integration. We will also provide a comparative analysis to study the evaluation metrics before and after integration. Furthermore, as discussed earlier, we will highlight the improvement in three core metrics.

6.2.1 Pre-Integration Performance Evaluation

Before integrating the Q-Learning agent with the Cowrie honeypot, we performed a stand-alone evaluation to assess its performance and the effectiveness of the learning agent. This evaluation focused on the key aspects of Agent Performance, Scalability and Robustness. These tests analysed metrics such as average reward, cumulative reward, Q-value updates, and policy convergence.

We evaluated the model's ability to optimise the system's responses to attack patterns by calculating reward trends over multiple episodes. The Q value analysis provided the effectiveness of the state action decision-making, verifying that the model was learning the best response strategies. Therefore, pre-integration tests were conducted to ensure that the MDP model was efficient and capable of supporting the integration with Cowrie for planned attack scenario simulations. Thus, this section discusses the performance metrics for the MDP model integrated with the Q-Learning algorithm prior to integration with Cowrie Honeypot.

Reward The result of the reward shows the following.

1. **Cumulative Reward:** The cumulative reward represents the total reward collected over 100 episodes, calculated by multiplying the average reward per episode by the total number of episodes. In our case, with an average reward of 290.45 per episode multiplied by 100 episodes, we obtained a final cumulative value of 29045.0. This high cumulative reward indicates that the MDP successfully achieved its optimisation objectives.

$$\text{Cumulative Reward} = \sum_{i=1}^{N} R_i$$

In this case, R_i is the total reward for the episode i, and N is the total number of episodes.

2. **Average Reward**: This value represents the reward per episode stands at 290.45. The value is obtained by dividing the cumulative reward by the number of episodes, which is 100. If the mean reward approximates the cumulative reward divided by the number of episodes, it reflects consistency in performance all over the episodes. The average reward calculated is **290.45**.

$$\text{Average Reward} = \frac{\text{Cumulative Reward}}{N}$$

In this case:

$$\text{Average Reward} = \frac{29045.0}{100} = 290.45$$

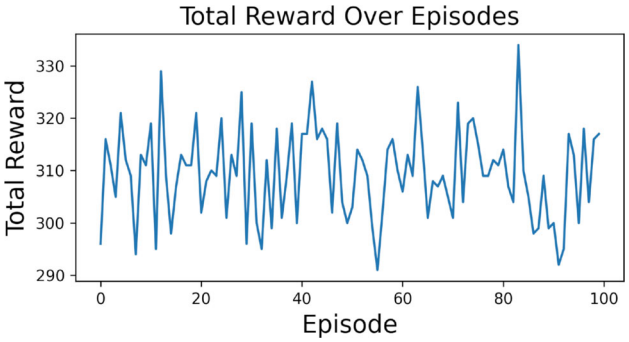

Fig. 3 Total rewards over episodes

3. **Rewards for each episode:** These values represent the rewards obtained for each episode. Figure 3 depicts the rewards received for each episode.
   - **Episode Variability:** The rewards for most episodes fall within an acceptable range of 290 and 340, indicating consistent model performance. While a few episodes show slightly lower rewards (e.g., 265), these are rare occurrences and can be considered outliers. Since these deviations are minimal and do not significantly impact the overall trend, the model's performance remains stable and reliable within the given range.
   - **Policy Improvement:** The MDP policy represents effective results given its high consistency and average reward. However, episodes with a lower range require policy improvement.
   - **Parameter Tuning:** The low standard deviation and lack of major trends represent learning parameters such as the learning rate and discount factor suitably tuned for this environment.

Analysing the metrics and their impacts contributes to understanding our MDP model's performance and applying improvements if necessary.

Q-Value This value represents an action-value where the agent obtains a reward through a specific action. Q-value donates Q(s, a) and suggests expected future reward for taking action [32]. Figure 4 and Figure 5 depict heatmaps for a detailed analysis for both before and after integration. This

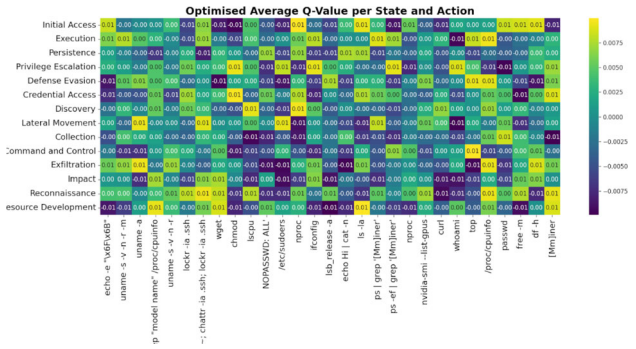

Fig. 4 Optimised average Q-Value for states and actions for pre-Integration

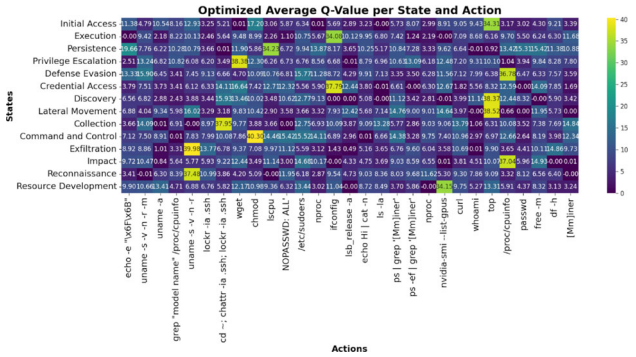

Fig. 5 Optimised average Q-Value for states and actions for post-Integration

allows us to gain insights into actions with higher q-values. In both heatmaps, darker colours, such as purple, indicate lower q-values, whereas brighter colours, such as yellow, represent higher q-values.

Policy A policy is a set of rules when deciding to take specific action to maximise the reward [32]. The policy analysis of our MDP model is visualised in Figure 3 accordingly. The total rewards graph represents a stable performance with a reward range between 290 and 340, reflecting effective policy. The spike points in the graph indicate episodes with distinguished performance; however, low points determine the potential for fine-tuning. The actions category distribution highlights that the policy often utilises "block" and "insult message" actions, emphasising threat mitigation and deterring strategies. The "delay" action indicates the strategy for extending the interaction and convincing the attacker to re-input alternatives. The "block" action, however, determines a defensive strategy to stop the attacker from further actions when harmful action is observed. Overall, all of these actions demonstrate an effective policy that contributes to integrating various tactics to maximise rewards in different attack scenarios. Figure 6 depicts these actions.

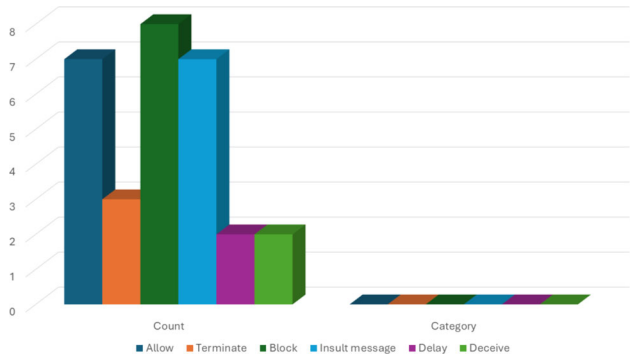

Fig. 6 Action category distribution

6.2.2 Post-Integration Performance Evaluation

In the second experiment, we evaluate the effectiveness of our Q-Cowrie's performance regarding dynamic behaviour, identifying new attack patterns, and learning stability. Analysing these metrics allows us to understand the model's conduct. The following discusses how these metrics play a role in Q-Cowri's performance evaluation.

Dynamic behaviour generation This metric contributes to Q-Cowrie's behaviour while interacting with attackers in the following contexts.

- **Resilience to Changes**: Resilience implies the ability of the MDP model to be effective in uncertain situations. For instance, if an attacker deploys a new tactic or technique that is new to Q-Cowrie, the model's resilience helps to manage and mitigate the threat while maintaining effective performance.
- **Adaptability**: This metric plays an important role in evaluating how quickly the MDP model can adapt to changes in a dynamic environment by optimising its policy. For example, the model learns the pattern of newly identified attacks and adjusts and optimises its policy to combat them effectively.
- **Dynamic Response Generation**: This metric enhances the Q-Cowrie to generate dynamic responses based on attackers' actions. For instance, if the attacker intends to download a malicious file through the curl -O http://malicious-site.com/malware.sh, or wget http://malicious-site.com/malware.sh, the Q-Cowrie honeypot decides whether to allow, terminate, delay, block, insult, or provide misleading information.

Identifying Unique Patterns The uniqueness of pattern identification plays an important role in evaluating the accuracy of the model. Our first experiment allowed us to identify common patterns of cryptomining and botnet attackers and apply this knowledge to our next cycle of experiments. For instance, by analysing the previous pat-

terns, the model knows that the common pattern for attackers is *"Initial Access"*>*"Execution"* >*"Privilege Escalation"*. However, unique patterns identify unusual state transitions, such as *"Initial Access"*>*"Privilege Escalation"*> *"Lateral Movement"* >*"Exfiltration"*. Moreover, launching specific commands (e.g. "terminate") in a specific state (e.g. "Persistence") triggers a high-risk action is happening. The MDP model, therefore, learns this as a new threat pattern and updates its policy to group it as a known attack. This allows the model to gain insights into effective decision-making and future mitigation strategies when identifying new patterns. Furthermore, the attacker may deploy a unique pattern to launch a process. For instance, deploying sophisticated tactics for downloading or installing a complex shell command.

Learning stability This metric measures how quickly the MDP model can learn new policies and adapt existing policies to new environmental changes. Therefore, the speed of adaptability and frequency of policy updates are sub-metrics used to evaluate this metric. The higher rate contributes to the speed with which the MDP model learns and updates its policy. For instance, in the case of newly identified attack patterns, a high rate of learning and optimising the policy interprets how fast the model can adapt itself to the new threats and, from there, can improve the defence mechanism.

7 Discussion

In this section, we discuss the key findings of our research and compare them with those of other studies. The primary objectives were to evaluate the performance of the Q-Cowrie MDP-based honeypot in identifying unique patterns, adaptability, and dynamic response generation. Identifying unique patterns supports improving the adaptability of our Q-Cowrie by learning the newly discovered attack patterns and identify the best set of dynamic responses to achieve highest reward. Moreover, a comparative analysis study promotes the strengths and potential flaws of our approach and offers an evaluation of our MDP model's effectiveness.

7.1 Key Findings

We identified several crucial insights, including identification of unique attack patterns, the MDP Model's learning stability, and the effectiveness of dynamic responses.

7.1.1 Unique Attack Patterns

After analysing the captured log files, we observed a noticeable spike in unknown and unique patterns and their associated frequencies post MDP integration. Figure 7 depicts the identified attack patterns pre and post MDP integration phases.

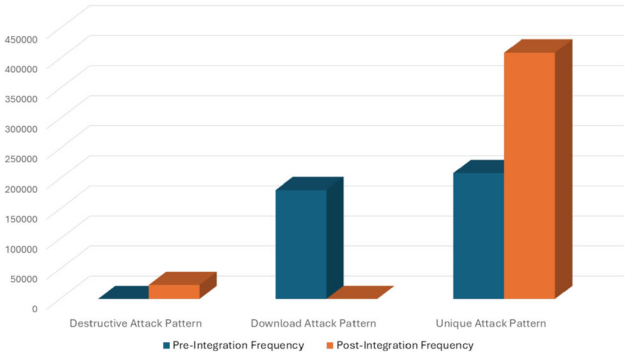

Fig. 7 Attack pattern identification pre and post MDP integration

Our observations revealed that most of the collected data contained malicious activities and destructive attacks. We define disruptive attacks as any malicious activity or attacker behaviour, such as executing malware or downloading harmful files.

The MDP integration improved threat detection in three ways:

- Enhanced identification of attacks with destructive patterns
- Improved collection and classification of unique and complex patterns
- Enabled more strategic responses based on learned attack data

We observed a significant increase in sophisticated download and execution attempts post-integration. This improvement in detection and response capabilities appeared to motivate attackers to develop more complex and unique methods to launch malicious scripts. To illustrate this evolution, we present below several examples of unusual attack patterns identified during our analysis phase.

Unique Pattern for Download and Execution Attempts In this instance, the attacker tried sophisticated forms of attack to ensure that the download or installation process was completely carried out. In this attack, the attacker first attempted to connect to a remote server on port 60107. Then, download a file using the "curl" or "wget" command. In case of failure of the download commands, the attacker simulates the HTTP request using "echo" manually. The attackers, therefore, make the file executable through "chmod". The encoded string "/tmp/bRgosFXgVP..." shows Base64-encoded or other obfuscation methods could be applied. This suggests that the executed script may include a payload or other commands to inject the malicious files and launch them.

Repeated Connection We also observed a unique pattern of repeated connections from specific IP addresses, with brief connections quickly closing often after some time. This trend

pattern has been present since the MDP integration. Table 5 illustrates the pattern of repeated connections from a single IP address with each session maintaining a consistent duration of 120 seconds. The repeated pattern suggests an automated script or brute-force attack, characterised by sessions consistently generated to maintain 120-second connections. This duration is notably long in this case, indicating a deliberate strategy to establish and maintain persistence before session termination.

SSH Client Versions Most connections use the SSH client version "SSH-2.0-libssh-0.2" with occasional entries for "SSH-2.0-makiko and SSH-2.0-Go". This could indicate different bots or tools attempting to interact with the server.

Login Command Executions Analysis revealed login attempts using common credentials (e.g., `root/1234 56`), indicating brute-force attacks. The attackers also executed system manipulation commands (`echo`, `cat`, `chmod`), combined with attempts to download and execute files, demonstrating a clear intention to deploy malware or malicious scripts.

7.1.2 MDP Learning Stability

Several metrics contribute to the learning stability of our MDP-based Cowrie. By discussing key performance metrics, the effectiveness of actions, and evaluating stability and adaptability, we gain comprehensive insight into our MDP model, enhancing the Q-Cowries' conduct. The following discusses these metrics.

Cumulative Reward This metric refers to the overall rewards obtained by MDP over multiple episodes. In our analysis, we discuss that the MDP model obtained a high cumulative reward of 29045.0, with an average reward of 290.45 per episode. These values demonstrate consistent performance across episodes. In addition, a value of 8.12 was assigned to the standard deviation, determining stability in the model learning process.

Actions Analysis This metric measures the effectiveness of decisions taken by the MDP model. The purpose of this paper is to identify the most effective strategies that enhance the success of the models. For example, as shown in Figure 8, the *Block* and *Insult Messages* were the most effective strategies reflecting the model response mechanism.

Model Stability The MDP model demonstrates stable performance with modern attack strategies. Moreover, our model can adapt to newly identified attack patterns by stabilising reward metrics after integrating new changes. Figure 8 shows the overall learning stability of the MDP model with respect to the state distribution, the response success rate, and the percentage of stability improvement to learn and adapt to dynamic changes.

- **State Distribution and Response Success (Left Diagram)**: Before MDP integration, the honeypot logged actions under the Recovery state at 80% with a minor rate captured under Command and Control (C&C) at about 20% and an insignificant activity under Credential Access below 20%. However, after MDP integration, the system records 80% for Recovery state, 15% for Execution, and about 10% for Persistence, with below 10% records for Credential Access, and no record for C&C. That means MDP-based decision-making enables the honeypot to have deeper interaction with attackers than simple scanning. Furthermore, the data recorded for Execution and Persistence states shows, most likely, that the attackers believe they are interacting with a real system rather than a honeypot. Moreover, the absence of attackers' activities under C&C may suggest that the attackers' engagement has shifted to direct interaction, likely due to the honeypot's responses during interaction.
- **Success Rate of Responses After MDP Integration (Right Diagram)**: After MDP integration, the honeypot recorded a high response rate in the Execution state at around 50%, followed by Persistence at approximately 47%, while Credential Access and Discovery showed only about 1% each. This means that the MDP-based honeypot provides deeper engagement when attackers execute commands or maintain access. The low rates in Credential Access and Discovery states suggest that attackers either attempted to have credential-based attacks or that the honeypot responded minimally during the initial reconnaissance stage to encourage further interaction.

The performance of honeypot shows the improvement of its adaptability when responding to attackers' actions. For instance, during the result analysis MDP-based honeypot shows the system can generate dynamic responses to attackers successfully at a rate of 94%. Additionally, its ability to learn from attackers and adjust its responses reflecting adaptability and learning shows the performance improvement of about 80%. Overall, the result discussed above represents the MDP-based (Q-Cowrie) honeypot is capable of interact intelligently and maintaining interaction to collect valuable behavioural data. The high response rate shows honeypot can maintain interaction with attackers, which is important for analysing attacks and intelligence gathering.

Performance Comparison This metric represents a performance comparison for our MDP model before and after integration. The results of the analysis show that the honeypot generated more diverse responses after integration. In addition, the attack patterns were collected more intelligently for the input of specific attackers, including textit "Miner","curl", and "wget", indicating cryptominer patterns.

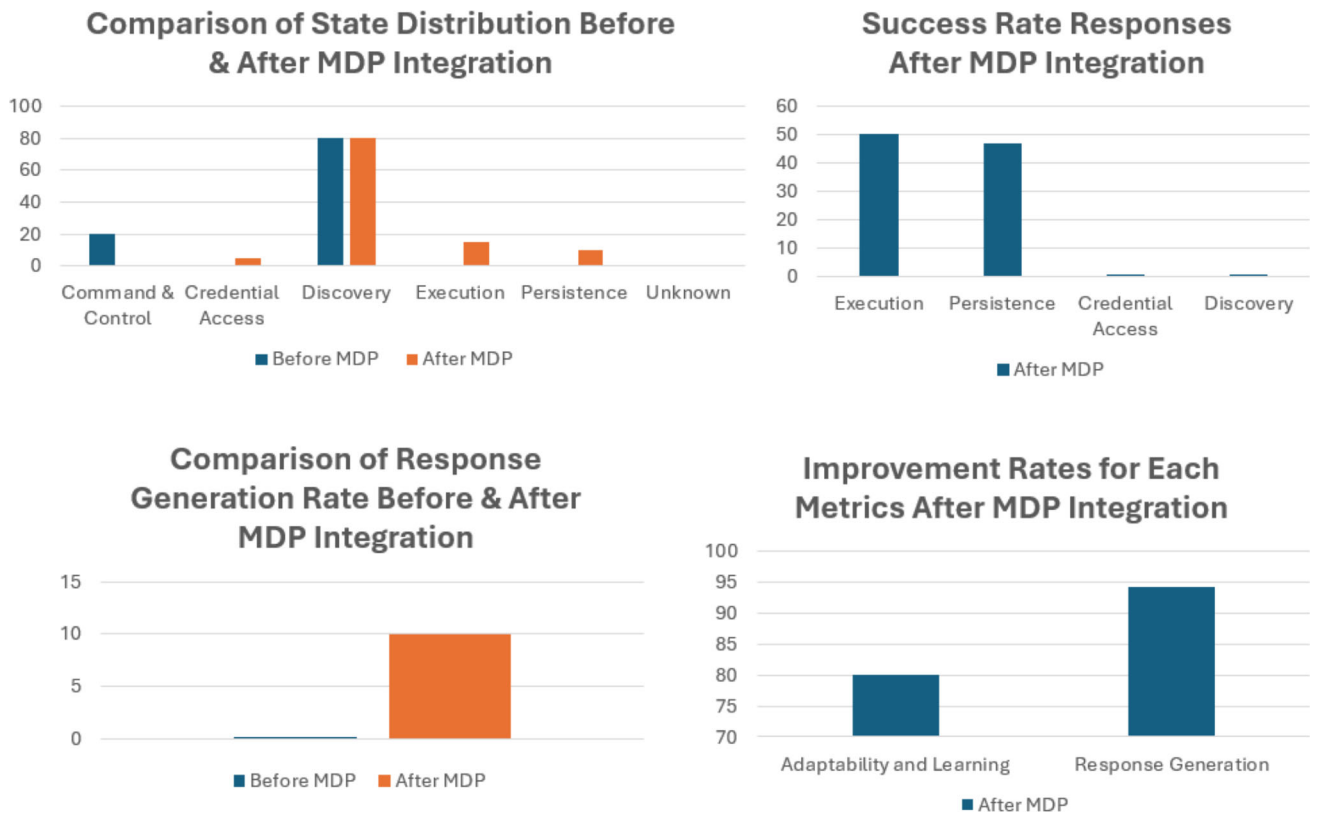

Fig. 8 MDP's model stability to adapt to new data

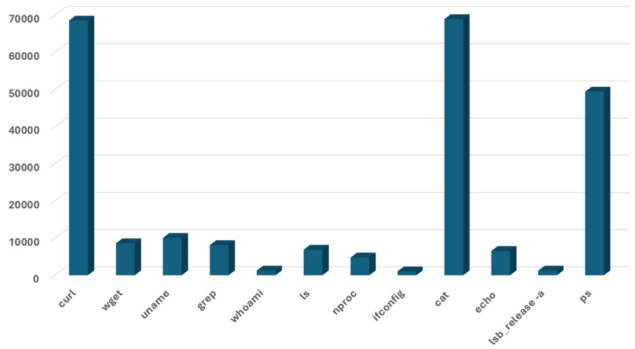

Fig. 9 Command category distribution for pre-MDP Integration

The following figures illustrate the various captured commands associated with their frequencies before and after MDP integration.

The results illustrated in Figures 9 and 10 show that the adapted configuration and integrated MDP honeypot were performed successfully. Figure 10 shows significant improvements in receiving expected input from the attacker, and the frequency numbers are evident for MDP's stability and effectiveness compared to Figure 9.

7.1.3 Dynamic Response Generation in Modified Cowrie (Experiment 1)

With the modified Cowrie honeypot configuration, the system does not possess autonomous decision-making ability. Therefore, when attackers execute malicious code or download malicious files, the responses are predefined and non-dynamic. For instance, when an attacker attempts to download a malicious file from "Malicious URL", Cowrie allows this to proceed. Due to this static configuration in response generation, attackers have more freedom to execute a larger number of destructive commands. Figure 9 depicts command deployments such as "curl and wget" spike at a frequency of approximately 68,000. Similarly, "cat and ps" commands are considerably higher, indicating attackers' attempts to read files and view the current active processes. Table 5 represents examples of Cowrie's responses retrieved from the admin's interface when attackers access malicious websites or download and execute scripts.

The details show the Cowrie honeypot, with no limitations or autonomous decision making, generates the "Allow" response to the attacker's malicious activity to be executed and saved on the system.

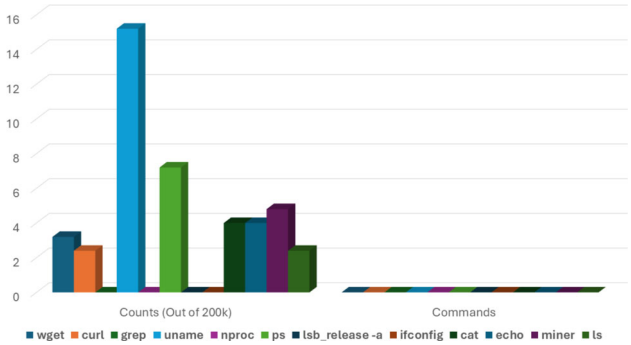

Fig. 10 Command category distribution post-MDP integration

Table 5 Cowrie "Allow" Responses to Malicious Activities Attempted by Attackers

Timestamp	Description of Activity and Q-Cowrie Response
2024-02-12 12:40:33	Attacker executed the command `wget <Malicious URL>`. Cowrie logged: `Command found: wget <Malicious URL>`.
2024-02-12 12:40:33	Cowrie simulated DNS resolution: `Resolving <Malicious URL> ... x.x.x.x`.
2024-02-12 12:40:33	Connection established to `< Malicious URL> (IP: x.x.x.x)`. Cowrie log entry: `Connecting to <Malicious URL> ... connected`.
2024-02-12 12:40:34	HTTP request sent; Cowrie simulated response: `HTTP request sent, awaiting response... 200 OK`.

7.1.4 Dynamic Response Generation in Q-Cowrie (Experiment 2)

Integration of the MDP model helps to autonomously improve response generation in the Q-Cowrie honeypot. The autonomous behaviour refers to the situation where Q-Cowrie can decide on the basis of its learning from past data. Analysis of the post-integration of the MDP model reveals significant changes. Q-Cowrie can autonomously decide to allow, block, delay, insult, deceive, or terminate attacker interactions as needed. The following analysis examines the attacker's inputs and their corresponding Q-Cowrie responses, specifically addressing three response types: *Allow*, *Block*, and *Delay*.

Allow This is Q-Cowrie's response when the attacker tries to download or access a non-malicious URL. Q-Cowrie understands this action as normal behaviour and thus allows it to be processed for further non-malicious actions. Allow response from Q-Cowrie can have different meanings to attackers.

1. **Testing Connectivity**: Attackers frequently test their Internet connectivity to validate that their commands are properly launched. Therefore, downloading a well-known, benign URL verifies the system's configuration and network settings operate as expected.
2. **Environment Reconnaissance**: The attackers intend to obtain information about the environment to devise future malicious activities.
3. **Avoiding Detection**: The attackers aim to remain covert and undetectable in the normal system and network behaviour to avoid being detected by defence mechanisms.
4. **Gaining Trust**: The attackers verify if Cowrie responds to their legitimate requests; this indicates that they can maintain access in the future.

Table 6 Q-Cowrie "Allow" Responses to Non-Malicious Activities Attempted by Attackers

Timestamp	Description of Activity and Q-Cowrie Response
2024-05-12 19:40:33	Attacker executed the command `wget <Non-Malicious URL>`. Q-Cowrie simulated execution and initiated a realistic download sequence.
2024-02-12 12:40:33	Q-Cowrie simulated network establishment: `Connecting to <Non-Malicious URL> ... connected`.
2024-02-12 12:40:33	HTTP request sent. Q-Cowrie responded with: `200 OK`. Response length unspecified. File saved as `/home/cowrie2/index.html`.
2024-02-12 12:40:34	File download completed successfully. Q-Cowrie logged: `/home/cowrie2/index.html [1,198,483 bytes]`.

Table 6 shows the Q-Cowrie response retrieved from the admin's interface when attackers aim to access non-malicious website. Q-Cowrie simulates a fully interactive shell environment that provides real-time feedback to attackers. For example, when the attacker executes the command "wget Non-Malicious URL", Q-Cowrie responds with realistic outputs, such as Connecting to "Non-Malicious URL... connected and 200 OK". These messages mimic the behaviour of a real system to keep attackers engaged and delay their realisation that they are inside a honeypot.

Block If Q-Cowrie detects malicious requests from attackers, the honeypot must decide. As provided in the sample response, Q-Cowrie knows that if the malicious URL is allowed, there will be a high negative reward due to the risk of compromise. However, due to its prevention strategy, Q-Cowrie blocked this malicious URL to gain a high positive reward. The rewards are the values used to guide the system's behaviour. Furthermore, block response is a decision that Q-Cowrie can take for the following reasons:

- *High Positive Reward for Blocking Access*
- *Maintaining Integrity*
- *Reducing Risk*
- *Intelligence Gathering*

The honeypot detects the attempt to access *Malicious URL* as a known malicious domain and blocks it for optimal decision-making. This leads the system to prevent compromise. However, if the honeypot allows this malicious URL, it leads the system to compromise, data theft, or further exploitation. Table 7 shows the Q-Cowrie response retrieved from the admin's interface when attackers aim to access a malicious website. In such situation, Q-Cowrie decides to block the attacker by simulating a DNS resolution failure, making it appear that the domain "Malicious URL" is not reachable. This block prevents the attacker from successfully downloading any malicious content while keeping them engaged in the system. The simulated response looks like a legitimate system error, preventing the attackers from realising they are interacting with a honeypot.

Delay If attackers aim to download or access a semi-malicious or malicious URL or file and encounter delaying in response, they will consider it as legitimate behaviour for one of the following reasons:

- **Mimicking Normal Behaviour**: Due to the network latency, server load and other factors, the system can respond slowly to upcoming requests. Therefore, attackers interpret it as normal behaviour and thus expect a delay in response when they interact with the honeypot.
- **Avoid Detection Strategies**: This is another reason the attackers are forced to slow down without alerting them to avoid significant losses. This can create a perception of the system's normal behaviour.
- **Evasion**: The delay response is a stealthy way to observe and monitor attackers' further behaviour. This may give the impression that the system is experiencing typical issues and thus attackers do not realise that they are being analysed.
- **False Sense of Security**: When attackers realise that their activities are not immediately blocked or flagged, they may perceive this as poor defence mechanisms or assume that their malicious actions are not detected and remain stealthy; hence, they are convinced to conduct harmful actions.

Table 7 Q-Cowrie "Block" Responses to Malicious Activities Attempted by Attackers

Timestamp	Description of Activity and Q-Cowrie Response
2024-05-12 19:39:07	Attacker executed the command `wget <Malicious URL>`. Q-Cowrie simulated connection initiation: `Connecting to < Malicious URL> : None... connected.` Followed by a blocked resolution sequence: `HTTP request sent, awaiting response... Resolving <Malicious URL>... failed: nodename nor servname provided, or not known. wget: unable to resolve host address '<Malicious URL>'.`
2024-05-12 19:39:08	Q-Cowrie retried DNS resolution and simulated repeated failure: `Resolving <Malicious URL>... failed: nodename nor servname provided, or not known. wget: unable to resolve host address' <Malicious URL>'.`

Table 8 shows the Q-Cowrie response retrieved from the admin's interface when attackers aim to access a malicious website.

Insult Message "insult" is a response mechanism triggered when attackers incorrectly input a command or execute particular commands. These mechanisms introduce a delay, disconnect the attacker, or generate a customised message to interrupt their workflow and frustrate their attempts. The primary goal of deploying this feature is to determine the attacker's technical skills when responding to erroneous or insulting messages. However, this feature was used in Heliza's [24] development to only uncover the attacker's identity by revealing the attacker's linguistic background. Figure 11 lists some examples of our insult messages.

7.1.5 Justification and Context for Using Insult Messages

Indeed, typical Linux-based systems do not employ provocative or humorous responses, and we acknowledge that the inclusion of insult messages could potentially alert attackers to the artificial nature of the honeypot environment. How-

Table 8 Q-Cowrie "Delay" Responses to Malicious Activities Attempted by Attackers

Timestamp	Description of Activity and Q-Cowrie Response
2024-05-12 19:37:32	Attacker executed the command `wget <Malicious URL>`. Q-Cowrie logged the event and initiated a simulated download process with intentional time delays to mimic network latency.
2024-05-12 19:37:32	Q-Cowrie simulated connection establishment: `Connecting to <Malicious URL>... connected.`
2024-05-12 19:37:32	HTTP request sent. Q-Cowrie introduced an artificial delay before responding: `Resolving <Malicious URL>... failed: nodename nor servname provided, or not known.`
2024-05-12 19:37:32	After completing the delay simulation, Q-Cowrie logged: `wget: unable to resolve host address '<Malicious URL>'.`

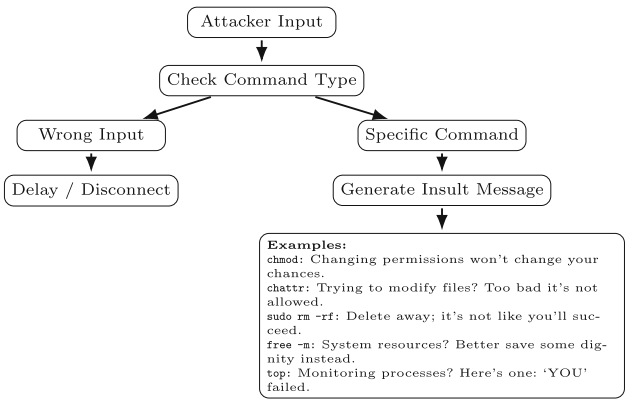

Fig. 11 Compact workflow of the insult message mechanism with example messages

ever, our motivation for including this specific feature comes from two main reasons:

1. **Background and Rationale from the Previous Work:** Our decision to implement insult messages was inspired by established honeypot research, notably the Heliza project [24], which effectively utilised provocative, human-like messages to encourage longer interactions and deeper engagement from human attackers. While our primary targets (cryptomining bots and automated systems) typically ignore these messages, given that bots are inherently unresponsive to emotional or provocative prompts, we included insult messages as an exploratory feature to help distinguish human attackers from automated systems. Human attackers, even if representing a small percentage (e.g., approximately 1%), might exhibit identifiable reactions to provocative responses, thus helping researchers better understand and differentiate human behaviour in automated attack environments.
2. **Addressing Realism and Risk of Exposure:** It is challenging to directly measure the risk of exposing a honeypot when using unusual responses like insult messages. Typically, researchers analyse indirect metrics, such as how long attackers stay connected and how many commands they run. We used these measures to see how insult messages affected attacker behaviour. While bots mostly ignore these messages, we included them as an experiment to help spot rare human attackers and gather more detailed data for future real-world use. We acknowledge the limitations of this method; however, it was implemented to enhance the honeypot's ability to distinguish human attackers from automated ones, based on previous studies.

7.2 Comparison with Previous Studies

This section compares proposed Q-Cowrie honeypot with other research studies such as HARM, Heliza and QRASSH in which RL and MDP based solutions have been adopted. The comparison was performed based on several features including the primary focus of each study, learning approach followed, adaptation frequency, autonomous capabilities, targeted threats, behavioural model, available use-case examples and educational application. The feature list covers both general behaviour aspects such as primary focus and technical concerns such as deployment methodologies.

Table 9 reports the comparison results. Differentiating our work from others, Q-Cowrie is focused on generating dynamic responses when attackers interact with this. The approach is based on Q-Learning to design MDP and offers continue adaptation. Specific threats Q-Cowrie deals with include cryptomining and Botnet attacks and this selection is based on attack data collection performed through a real-world experiment (Experiment 1) rather than relying on assumptions of attacks popularity. From existing studies, HARM also similarly used Q-Learning along with SARSA agents for periodic adaptability to malware behaviour [39]. QRASSH is designed to address SSH-based threats through Deep Q-Learning [11]. Although research studies reported in Table 9 are diverse in many aspects such as development methodologies, data collection and evaluation, but generally these studies share same goal to adapt and learn attackers' behaviour.

Comparing Q-Cowrie with other studies in terms of methodological validation along with other aspects such as

Table 9 Comparison of Q-Cowrie with Existing Approaches

Feature	HARM	Heliza	QRASSH	Q-Cowrie
Primary focus	Automated and repetitive malware	Profiling attackers	SSH-related threats	Detection and response generation
Learning approach	Periodic policy updates	Markov chain	Deep Q-learning (DQN)	Q-Learning
Adaptation frequency	Periodic	Continuous	Real-time	Continuous
Autonomicity degree	Autonomous	Autonomous	Autonomous	Autonomous
Target threats	Malware-focused	General attack profiling	SSH-specific threats	Cryptomining and botnets
Behavioural model	Q-Learning and SARSA	Markov chain	Deep Q-learning (DQN)	Q-Learning modeling an MDP
Use-case example	Periodically adapt to evolving malware patterns	Profiling attacker behaviour for analysis	Defend and study SSH-related attacks	Dynamically respond to sophisticated attack patterns
Educational application	Improve automated-malware handling	Awareness and basic threat detection	Learn and understand attacker SSH behaviours	Expose unique patterns and enable dynamic responses

complexity, data availability and human interpretability are interesting dimensions to explore. Section 8 further explains these aspects in detail and proposes some future directions.

7.3 Practical Applicability of Our Proposed Approach

Our proposed Q-Cowrie system shows strong potential for practical application in real-world environments where defenders encounter increasingly sophisticated threats. By dynamically adapting responses to attacker behaviours, Q-Cowrie can extend the interaction with the attacker, enabling security teams to collect richer threat intelligence and gain deeper insights into evolving attack strategies, including botnet and cryptomining campaigns. Although this study demonstrates the effectiveness of the system in the real-world environment in which we controlled the interaction, future work will focus on refining the MDP model to improve prediction accuracy, extending the scope of target threats to include phishing and ransomware, and addressing practical challenges such as training complexity and data quality. These directions aim to close the gap between experimental research and operational deployment in dynamic real-world environments.

8 Limitations and Future Work

Implementing an integrated MDP model with a Cowrie honeypot contributes to a significant improvement in adaptability, learning, and dynamic response generation. However, several challenges and limitations were encountered during this research. The following discussion identifies limitations related to model complexity, computational demands, scalability, and practical deployment.

Learning Model and Computational Complexities Training the MDP-based learning model is computationally intensive due to the progressive learning process in RL algorithms. As the state-action space expands, the processing time and hardware requirements increase significantly, limiting the scalability in larger or distributed honeypot environments. Therefore, proper hardware and software configurations play an important role in ensuring efficient model training and execution. Moreover, integrating new data into the model while maintaining accuracy can increase computational complexity.

Data Integrity and Availability High-quality, diverse, and accurate datasets are essential to develop an effective model. Noisy or incomplete data can significantly reduce the adaptability and self-learning of the MDP model, limiting its ability to manage diverse attack patterns and evolving threat environments.

Adaptability and Detectability While the integrated MDP-Cowrie system enhances dynamic response generation, maintaining adaptability against modern threats remains challenging. Continuous updates and maintenance are required to ensure effective integration when adapting to the latest threat intelligence. In addition, advanced attackers may identify Q-Cowrie's behavioural patterns through extended interaction or fingerprinting analysis, exposing the Q-Cowrie to detection. Therefore, improving evasion and minimising detectability remain persistent challenges for future development.

Human and Interpretability Factors Designing an MDP model demands in-depth knowledge of RL algorithms, data analysis and cybersecurity. However, the decision-making process of the MDP model can be complex and difficult for humans to interpret, potentially reducing transparency. Therefore, using visualisation tools or explainable AI techniques to analyse MDP-based responses can enhance operational usability and interpretability.

Future Work Future research can address these limitations through the integration of Deep Reinforcement Learning (DRL) to enable complex policy learning and improve scalability. The design of a hybrid honeypot architecture that integrates low and high interaction layers can achieve an optimal balance among realism, safety and resource efficiency. Furthermore, coupling Q-Cowrie with Intrusion Detection Systems (IDS) and Security Information and Event Management (SIEM) solutions can strengthen situational awareness and automate the correlation between honeypot data and network defence mechanisms.

9 Conclusion

In conclusion, our Q-Cowrie honeypot demonstrated the effectiveness of integrating MDP-based models and RL techniques. Our approach significantly enhanced the identification of unique attack patterns, improved adaptability and enabled dynamic response generation for cryptomining and botnet attack scenarios. Our MDP models based on real-world attack data helped us develop effective responses to attacker actions and behaviours. Limitations of our system such as adaptability to new or zero-day attacks and challenges with learning model complexities associated with hardware and software are also discussed. Other interdisciplinary avenues such as exploring criminal ecosystems and customising the honeypots to other specific real-world threat profiles can be further explored in future research.

Author Contributions Maryam Var Naseri developed main ideas, collected data, proposed methodology, conducted experiments, performed analysis, wrote the original draft of the paper, reviewed and edited it. Ian Welch supervised the work, provided guidance to improve ideas and methodology, reviewed and edited the paper. Junaid Haseeb provided guidance to improve ideas and methodology, reviewed and edited the paper. Masood Mansoori supervised the work, provided guidance to improve ideas and methodology, reviewed and edited the paper.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

Data Availability The datasets generated and/or analyzed during the current study are available from the corresponding author upon request.

Declarations

Conflicts of Interest The authors do not have any financial or non-financial interests to declare that could have appeared to influence the work reported in this paper.

Competing interests The authors declare no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Yuen, J.: Automated cyber red teaming. Cyber And Electronic Warfare Division, Defence Science And Technology Organisation, Edinburgh South Australia, Australia, Tech. Rep. (2015)
2. Li, Y., Liu, Q.: A comprehensive review study of cyber-attacks and cyber security; Emerging trends and recent developments. Energy Rep. **7**, 8176–8186 (2021)
3. Javadpour, A., Ja'fari, F., Taleb, T., Shojafar, M., Benzaïd, C. A.: comprehensive survey on cyber deception techniques to improve honeypot performance. Computers & Security. pp. 103792 (2024)
4. Ahmad, W., Raza, M., Nawaz, S., Waqas, F.: Detection and analysis of active attacks using honeypot. International Journal Of Computer Applications. **184**, 27–31 (2023)
5. Lanka, P., Gupta, K., Varol, C.: Intelligent threat detection-AI-driven analysis of honeypot data to counter cyber threats. Electronics **13**, 2465 (2024)
6. Zakaria, W., Kiah, M.: A review of dynamic and intelligent honeypots. ScienceAsia **39**, 1–5 (2013)
7. Liu, S., Feng, P., Cao, J., He, X., Chin, T., Sun, K., Li, Q.: Consistency is All I Ask: Attacks and Countermeasures on the Network Context of Distributed Honeypots. International Conference On Detection Of Intrusions And Malware, And Vulnerability Assessment. pp. 197-217 (2022)
8. Huang, Y., Huang, L., Zhu, Q.: Reinforcement learning for feedback-enabled cyber resilience. Annu. Rev. Control. **53**, 273–295 (2022)
9. Franco, J., Aris, A., Canberk, B., Uluagac, A.: A survey of honeypots and honeynets for internet of things, industrial internet of things, and cyber-physical systems. IEEE Communications Surveys & Tutorials. **23**, 2351–2383 (2021)
10. Mohammadzadeh, H., Mansoori, M., Welch, I.: Evaluation of fingerprinting techniques and a windows-based dynamic honeypot. Proceedings Of The Eleventh Australasian Information Security Conference-Volume **138**, 59–66 (2013)
11. Pauna, A., Iacob, A., Bica, I.: Qrassh-a self-adaptive ssh honeypot driven by q-learning. 2018 International Conference On Communications (COMM). pp. 441-446 (2018)

12. Gutierrez, M., Kiekintveld, C.: Bandits for Cybersecurity: Adaptive Intrusion Detection Using Honeypots. Artificial Intelligence For Cyber Security, AAAI Workshop (2016)
13. Fan, W., Du, Z., Fernández, D., Villagrá, V.: Enabling an anatomic view to investigate honeypot systems: A survey. IEEE Syst. J. **12**, 3906–3919 (2017)
14. Thom, J., Shah, Y., Sengupta, S.: Correlation of cyber threat intelligence data across global honeypots. 2021 IEEE 11th Annual Computing And Communication Workshop And Conference (CCWC). pp. 0766-0772 (2021)
15. 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. **12**, 1508 (2024)
16. Li, Y.: Reinforcement learning in practice: Opportunities and challenges. ArXiv Preprint ArXiv:2202.11296. (2022)
17. Ding, Z., Dong, H.: Challenges of reinforcement learning. Deep Reinforcement Learning: Fundamentals, Research And Applications. pp. 249-272 (2020)
18. Dowling, S., Schukat, M., Barrett, E.: New framework for adaptive and agile honeypots. ETRI J. **42**, 965–975 (2020)
19. Puterman, M.: Markov decision processes: discrete stochastic dynamic programming. (John Wiley & Sons,2014)
20. Ding, Z., Huang, Y., Yuan, H., Dong, H.: Introduction to reinforcement learning. Deep Reinforcement Learning: Fundamentals, Research And Applications. pp. 47-123 (2020)
21. Guan, C., Liu, H., Cao, G., Zhu, S., La Porta, T.: HoneyIoT: Adaptive High-Interaction Honeypot for IoT Devices Through Reinforcement Learning. Proceedings Of The 16th ACM Conference On Security And Privacy In Wireless And Mobile Networks. pp. 49-59 (2023)
22. López, P., Pérez, M., Nespoli, P.: Cyber Deception: State of the art, Trends and Open challenges. ArXiv Preprint ArXiv:2409.07194. (2024)
23. Abdou, A., Sheatsley, R., Beugin, Y., Shipp, T., McDaniel, P.: HoneyModels: Machine learning honeypots. MILCOM 2021-2021 IEEE Military Communications Conference (MILCOM). pp. 886-891 (2021)
24. Wagener, G., State, R., Dulaunoy, A., Engel, T.: Heliza: talking dirty to the attackers. J. Comput. Virol. **7**, 221–232 (2011)
25. Dowling, S., Schukat, M., Barrett, E.: Using reinforcement learning to conceal honeypot functionality. Machine Learning And Knowledge Discovery In Databases: European Conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018, Proceedings, Part III 18. pp. 341-355 (2019)
26. Navarro Ferrer, O.: Analysis of reinforcement learning techniques applied to honeypot systems. (Universitat Oberta de Catalunya (UOC) (2021)
27. Hayatle, O., Otrok, H., Youssef, A.: A markov decision process model for high interaction honeypots. Information Security Journal: A Global Perspective. **22**, 159–170 (2013)
28. Pashaei, A., Akbari, M., Lighvan, M., Charmin, A.: Early Intrusion Detection System using honeypot for industrial control networks. Results In Engineering. **16**, 100576 (2022)
29. Cabral, W., Valli, C., Sikos, L., Wakeling, S.: Review and analysis of cowrie artefacts and their potential to be used deceptively. 2019 International Conference On Computational Science And Computational Intelligence (CSCI). pp. 166-171 (2019)
30. Hetzler, C., Chen, Z., Khan, T.: Analysis of ssh honeypot effectiveness. Future Of Information And Communication Conference. pp. 759-782 (2023)
31. Xiang, X., Foo, S.: Recent advances in deep reinforcement learning applications for solving partially observable markov decision processes (pomdp) problems: Part 1-fundamentals and applications in games, robotics and natural language processing. Machine Learning And Knowledge Extraction. **3**, 554–581 (2021)
32. Doltsinis, S., Ferreira, P., Lohse, N.: An MDP model-based reinforcement learning approach for production station ramp-up optimization: Q-learning analysis. IEEE Transactions On Systems, Man, And Cybernetics: Systems. **44**, 1125–1138 (2014)
33. Huang, L., Zhu, Q.: Adaptive honeypot engagement through reinforcement learning of semi-Markov decision processes. International Conference On Decision And Game Theory For Security. **196–216** (2019)
34. Suratkar, S., Shah, K., Sood, A., Loya, A., Bisure, D., Patil, U., Kazi, F.: An adaptive honeypot using Q-learning with severity analyzer. Journal Of Ambient Intelligence And Humanized Computing. **13**(10), 4865–4876 (2022)
35. Alavizadeh, H., Alavizadeh, H., Jang-Jaccard, J.: Deep Q-learning based reinforcement learning approach for network intrusion detection. Computers. **11**(3), 41 (2022)
36. Amin, M., Othman, M.: Re-exploration of ε-greedy in deep reinforcement learning. RiTA 2020: Proceedings Of The 8th International Conference On Robot Intelligence Technology And Applications. pp. 264-272 (2021)
37. Kaloev, M., Krastev, G.: Tailored Learning Rates for Reinforcement Learning: A Visual Exploration and Guideline Formulation. 2023 7th International Symposium On Innovative Approaches In Smart Technologies (ISAS). pp. 1-7 (2023)
38. MITRE, T. ATT&CK Matrix for Enterprise. Available at:. (https://attack.mitre.org/), Accessed: September 10 (2024)
39. Dowling, S., Schukat, M., Barrett, E.: Improving adaptive honeypot functionality with efficient reinforcement learning parameters for automated malware. Journal Of Cyber Security Technology. **2**, 75–91 (2018)
40. Khan, W., Khan, M., Muhaya, F., Aalsalem, M., Chao, H.: A comprehensive study of email spam botnet detection. IEEE Communications Surveys & Tutorials. **17**, 2271–2295 (2015)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.