ECTI Transactions on Computer and Information Technology
Journal homepage: https://ph01.tci-thaijo.org/index.php/ecticit/
Published by the ECTI Association, Thailand, ISSN: 2286-9131

Intelligent Honeypot for Web Applications: Leveraging Seq2Seq and Reinforcement Learning

Ananya Varadarajan[1], Ashwin Chandrasekaran[2], Rachana Binumohan[3], Rahul Huliyar Ravishankar[4] and Gokul Kannan Sadasivam[5]

ABSTRACT

An intelligent honeypot system designed to mimic legitimate websites using Sequence-to-Sequence (Seq2Seq) learning and Deep Q-Learning. The system generates realistic, contextually appropriate responses to attacker queries, prolonging interactions and providing insights into malicious behaviors while safeguarding actual systems. The Seq2Seq model, trained on HTTP request-response pairs, enables the honeypot to produce responses that closely resemble those of real servers, enhancing its ability to deceive attackers. Deep Q-Learning optimizes engagement by selecting the most effective responses through a custom reward function, balancing realism and interactivity to maximize session length. Performance was evaluated using metrics such as Response Realism Rate (RRR), Semantic Consistency Accuracy (SCA), and Average Session Length (ASL). The honeypot achieved an RRR of 92.3%, an SCA of 89.7%, and a 94.5% Optimal Response Selection Rate (ORSR). These advancements increased ASL by 143.5%, from 3.2 to 7.8 exchanges, reflecting prolonged attacker engagement. By integrating Seq2Seq and Deep Q-Learning, this honeypot demonstrates significant improvements in generating convincing responses and sustaining interactions. These results contribute to modern cybersecurity by providing a practical and theoretical framework for developing next-generation honeypots capable of deceiving attackers and gathering actionable intelligence.

Article information:

Keywords: Cybersecurity, Deep Q-Learning, Honeypots, Seq2Seq

Article history:
Received: May 20, 2025
Revised: November 20, 2025
Accepted: February 19, 2026
Published: February 28, 2026 (Online)

DOI: 10.37936/ecti-cit.2026202.262099

1. INTRODUCTION

A Honeypot is one of the cyber deception technologies, it is a vulnerable system that is designed to be attacked and compromised [1]. They help organizations gather intelligence on attack methods, adversary motives, and emerging threats. Unlike conventional cybersecurity solutions that passively monitor or block threats, honeypots enable defenders to study attacks in real time, analyze malicious behaviors, and improve security measures based on observed techniques [2]. As early as 1989, the concept of honeypots was proposed. Researchers mostly used honeypots for small-scale research in the past [3]. After years of development, honeypots have become a powerful tool to gather information on threats. At the same time, in 2024, cyber-attacks occur at an alarming rate of approximately *2,244* times daily, which translates to a breach every 39 seconds [4] and the common attack vectors mention that Phishing is the most prevalent attack method, accounting for over *80%* of reported security incidents, while SQL injection and command injection attacks are among the top ten most common types of attacks [5].

For many businesses and organizations, it is becoming increasingly crucial to identify cyber assaults before they compromise unpatched or otherwise unprotected web services. One particularly valuable tool for collecting intelligence on such attacks is a honeypot—a decoy system intentionally exposed to potential attackers in order to observe and analyze their behavior. Honeypots are capable of recording advanced and previously unknown threats, including zero-day attacks [6], making them an essential part of proactive cybersecurity strategies. Traditional network firewalls, though widely implemented, often fall

[1,2,3,4,5]The authors are with the Department of Computer Science Engineering, PES University, 560100, Bengaluru, India, Email: pes2202100832@pesu.pes.edu, pes2202100749@pesu.pes.edu, pes2202100742@pesu.pes.edu, pes2202100378@pesu.pes.edu and gokul@pes.edu
[2]Corresponding author: ashwin.chandra08@gmail.com

short in preventing web application attacks because these attacks occur at the application layer of the TCP/IP model. To address this gap, Web Application Firewalls (WAFs) were developed to detect and block threats specifically targeting that layer. Despite their usefulness, attackers are constantly evolving their techniques. They frequently employ Web Application Vulnerability Scanners (WAVs) or develop custom scripts designed to bypass WAFs. Although rule-based WAFs can be effective against WAVs, they struggle significantly when confronted with zero-day exploits, uniquely crafted payloads, or sophisticated custom scripts. In response, security researchers have increasingly turned to deep learning techniques [7] and advanced machine learning algorithms [8], which are capable of identifying anomalies with impressive accuracy. These AI-driven methods offer a marked improvement over traditional, rule-based systems. However, it's important to note that most of these solutions are defensive in nature, focusing primarily on detecting and preventing web application threats rather than engaging with them. In recent years, there has been a growing interest in gaining a deeper understanding of attacker behavior, tactics, techniques, and overall capabilities [9].

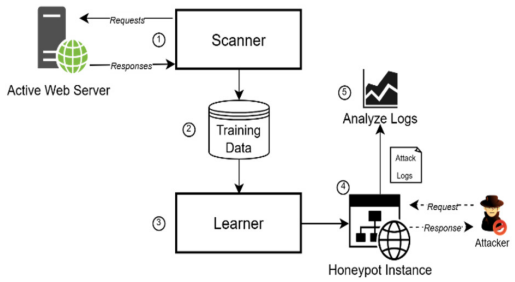

Fig.1: *A high-level overview.*

This research delves into the design and practical implementation of an intelligent honeypot tailored specifically for web applications. It leverages the power of Sequence-to-Sequence (Seq2Seq) models in combination with Reinforcement Learning to create a dynamic system capable of not only detecting but also interacting with and analyzing malicious behavior in real time. This approach aims to bridge the gap between traditional defensive systems and more adaptive, intelligence-driven cybersecurity solutions.

2. BACKGROUND AND RELATED WORKS

Ensuring the security of web applications is paramount, given their role in handling sensitive user information such as financial details and personally identifiable information (PII). Cybercriminals often exploit vulnerabilities within these applications to gain unauthorized access, leading to significant consequences including identity theft, financial fraud, and large-scale data breaches. To combat these threats, organizations employ various security mechanisms, ranging from preventive measures like encryption and access controls to active defense strategies such as honeypots and intrusion detection systems (IDS) [10].

While Intrusion Detection Systems (IDS) take a reactive approach, by detecting and alerting administrators about ongoing intrusions after they occur [11], Honeypots serve as proactive defense mechanisms designed to attract and engage attackers, allowing organizations to monitor malicious activities, analyze attack patterns, and develop robust cybersecurity strategies [1].

They are mainly classified into two categories based on the level of interaction: high-interaction honeypots and low-interaction honeypots. Our research is categorized into another interaction level, intelligent interaction honeypots.

2.1 Low Interaction Honeypot

Low interaction honeypots are designed to mimic network devices or services, but they do not fully replicate the internal workings or complex behaviors of real systems. Instead, they offer a limited set of responses that resemble those of the intended device or service, creating a surface-level illusion of a functioning system. These honeypots typically emulate only the essential network interactions, avoiding the complexities of a fully operational environment. Their responses to requests typically resemble those of the intended device. For instance, the Honeyd [12] honeypot utilizes a bash script to manage incoming requests, returning a standard "404 not found" unless explicitly programmed to recognize and respond to specific requests.

Since low interaction honeypots only provide basic responses, skilled attackers can often recognize them by sending unexpected or malformed requests and analyzing the responses. If the honeypot fails to behave exactly like a real device, an attacker may be able to identify it as a trap. This fingerprinting process can render the honeypot less effective, as attackers may choose to avoid interacting with it or use evasion techniques to bypass detection.

2.2 High Interaction Honeypot

High interaction honeypots provide attackers with access to a fully operational system, closely resembling a real network or application environment. Unlike low interaction honeypots, which offer only limited responses and basic simulations. These types of honeypots gather more sophisticated information about cyberattacks. The experiment described in [13] deals with high-interaction web honeypots configured inside the virtual machines running Linux OS and programmed using PHP web applications. This particular honeytrap is resource-consuming, complicated, and may take a while for deployment and upkeep. The scalability is thus reduced with an increase in the number of such honeypots.

2.3 Intelligent Interaction Honeypot

The first intelligent interaction honeypot was introduced in 2017 on IoTCandyJar [14]. Intelligent-interaction honeypots are a kind of interactive way of using an attacker to improve the chances of capturing the attacker's actions rather than imitating accurately the behavior of a targeted service or device, as in high, low, or hybrid interactions [15]. Intelligent interaction honeypots have mostly the use of machine learning algorithms for analyzing and adapting to evolving attack patterns, allowing them to dynamically engage with potential threats on real-time interaction. In this project, the intelligent interaction honeypot is extended to focus on web applications, given their widespread use and high vulnerability to common attack techniques like SQL Injection and Command Injection.

3. METHODOLOGY

3.1 High-level Overview

Fig. 1. provides a conceptual outline of the intelligent honeypot system. The *scanner* module detects active web servers with input fields on the internet, collecting HTTP request-response pairs that are later utilized as training data for the *learner* module. The honeypot instance, deployed on the internet, functions as the interface of the machine learning model, facilitating the collection of attack logs generated by malicious actors. These logs are systematically analyzed and visualized on a dedicated dashboard, providing valuable insights for security professionals. For instance, the dashboard displays statistical summaries of incoming requests and details regarding the attacker's session length.

3.2 Scanner Module

The *scanner* locates active, publicly accessible websites on the internet and collects a dataset of HTTP request-response pairs by targeting input fields with crafted attack payloads to uncover potential vulnerabilities. The gathered data is stored in a MySQL database and serves as training material for the honeypot's machine learning (ML) model. This process allows the honeypot to emulate authentic web interactions, producing responses that closely resemble those of legitimate web services and minimizing the chances of detection by attackers.

The *scanner* module automates data collection using Selenium, a browser automation tool that mimics human behavior, making it ideal for tasks such as filling out forms, submitting login credentials, clicking buttons, and navigating through web pages [16]. The login component inputs crafted attack payloads as username and password into the forms and clicks the submit button. Chrome DevTools Protocol (CDP) captures detailed HTTP request and response data.

Table 1: *Example of an HTTP Request Table.*

method	url	payload	headers	res_id
GET	/index.jsp	{}	Acce...	1
POST	/login.jsp	OR1′=...	Conn...	2

Table 2: *Example of an HTTP Response Table.*

res_id	status_code	headers	body
1	200	Content...	<html>Welc...
2	401	Last-m...	<p>Incorrect...

CDP provides low-level access to Chrome's developer tools, allowing for granular control over network traffic, page rendering, and browser behavior. Manual payload submission took *35* seconds per payload, while automation reduced it to 7 seconds. With 5 concurrent threads, processing 15 payloads took just 21 seconds—a *96%*-time reduction, making automation essential for scalable data collection.

Over time, this results in a dataset that grew to over 100,000 rows of data, containing a mixture of GET and POST methods, including SQL injection, command injection, and brute-force attempts. To quantify diversity, we computed category counts and distribution (brute-force 59%, SQLi 19%, command injection 14%, other/legitimate 8%) and used stratified sampling during training to maintain representativeness across categories.

All data collection was conducted under a research-only policy and adhered to strict safeguards. Scanner activity was rate-limited and scheduled with randomized intervals to prevent any denial-of-service–like effects. Only non-destructive probes were used throughout the study; payloads were constrained to benign, controlled patterns intended solely to elicit server responses without altering application state. Endpoints known to be sensitive or private, for example, those requiring multi-factor authentication or corporate-restricted access, were explicitly excluded from all scans. Each request–response pair underwent automated anonymization to ensure that no sensitive information was retained and passed through a standardized preprocessing workflow that included tokenization of HTTP components (method, path, headers, and body), normalization of whitespace and character encodings, and canonicalization of variable fields such as timestamps, session identifiers, and large numeric values through deterministic token substitution.

The MySQL database is structured with two dedicated tables to efficiently store and manage the collected HTTP requests and their corresponding responses. An example of the correspondence between requests and responses used in the learner module is shown in Table 1. This table is responsible for storing incoming HTTP requests, maintains essential details such as the HTTP method (e.g., GET, POST), the target URL of the website being accessed, the payload

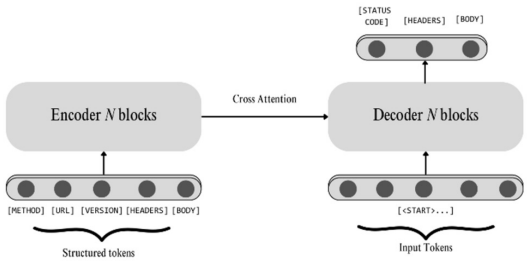

Fig.2: *Seq2Seq Transformer Architecture powering the Learner Module.*

submitted in the request body, and the HTTP headers accompanying the request. Additionally, each request entry is assigned a unique response identifier (res_id) that establishes a direct mapping between a request and its corresponding response. Table 2, which records HTTP responses, is designed to store the status code returned by the server (e.g., 200 OK, 404 Not Found, 500 Internal Server Error), the HTTP headers present in the response, and the actual response body containing the content returned to the requester. The relationship between these two tables is maintained using the res_id, ensuring that each request can be linked to a single, specific response. Our honeypot will generate a similar structure of responses to mimic a legitimate website.

3.3 Learner Module

The learner module learns the ability to generate proper responses to attack requests leveraging the Seq2Seq transformer model and Deep Q-Learning (DQL). The Seq2Seq Transformer model, shown in Fig. 2, generates realistic HTTP responses by simulating web application behavior through an encoder-decoder architecture [17]. Raw HTTP requests are pre-processed using a custom tokenizer that segments the stream into discrete semantic units: Method (GET/POST), Endpoint (URL), Protocol Version, Headers, and Body. To handle the high variability of attack payloads, we utilized a Byte-Pair Encoding (BPE) tokenizer with a vocabulary size of 32,000 tokens. This allows the model to efficiently represent rare strings found in, for example, SQL injection and XSS payloads without bloating the embedding space.

The encoder consists of N stacked blocks with six layers in the configuration used for this work. Each block utilizes eight-head multi-head self-attention to analyse dependencies between request headers and the body. For example, the attention mechanism learns to associate a "Content-Type: application/json" header with the structural expectations of the JSON body, creating a context-aware latent representation.

The decoder mirrors the encoder structure but employs Masked Self-Attention to prevent the model from "seeing" future tokens during training. In addition, each decoder layer includes a cross-attention module that attends to the encoder's output representations. This design allows the model to condition its generated response on the specific characteristics of the incoming request or potential attack vector.

The encoder processes incoming HTTP requests (e.g., POST /login HTTP/1.1 with JSON credentials) by first tokenizing the request into components such as the method, path, headers, and body. These tokens are mapped to numerical embeddings augmented with positional encodings to preserve sequential order. Multi-head self-attention mechanisms then analyze dependencies across the request, such as the relationship between headers (e.g., Content-Type: application/json) and the payload (e.g., {"username": "admin"}), transforming them into context-aware latent representations. This enables the model to capture nuanced interactions, such as how a malformed Content-Length header might invalidate the body structure.

The decoder leverages these representations to autoregressively generate coherent, contextually aligned HTTP responses (e.g., HTTP/1.1 200 OK<html>...). Using masked self-attention, the decoder ensures each predicted token (e.g., status codes, HTML content) depends only on prior tokens, maintaining syntactic validity. Simultaneously, cross-attention layers align the response with the encoder's request context, preventing inconsistencies like responding 404 Not Found to a valid POST /login. The model's ability to handle variable-length sequences and long-range dependencies allows it to generalize across diverse HTTP interactions, from simple GET requests to complex API calls with nested JSON payloads. This makes it particularly valuable for applications requiring dynamic traffic generation, such as penetration testing, API sandboxing, or load-testing frameworks.

Beam search is employed during the decoding phase to enhance response quality by considering multiple candidate sequences rather than making decisions based solely on immediate token probabilities. Unlike greedy decoding, which selects the most probable token at each step without considering long-term coherence, beam search maintains a fixed number of top-k candidate sequences (beams) at each decoding step.

This approach allows for a more globally optimal response selection, as it evaluates sequences based on their cumulative probability rather than making locally optimal choices that may lead to suboptimal final responses [18]. At each step, top-k tokens are chosen based on probabilities, and sequences are extended iteratively until completion or maximum length is reached. The final response is selected as the sequence with the highest cumulative score using equation (1):

$$\sum_{t=1}^{T} \log P\left(\frac{y_t}{y_{<t}}, X\right) \quad (1)$$

Beam search can be improved in a number of ways to further improve the caliber of the generated responses. Length normalization improves the selection of coherent responses by ensuring that longer sequences are not unjustly penalized in comparison to shorter ones. By encouraging the model to focus on different portions of the input rather than overemphasizing particular tokens, coverage penalties help reduce the problems caused by repetition. Furthermore, beam search can be combined with stochastic sampling to add controlled variability in response generation, which will make honeypot interactions seem less predictable and more natural to attackers. Together, these improvements increase response diversity, which increases the system's ability to trick and interact with adversaries.

Fig.3: *Q-values of actions for chosen response.*

The Seq2Seq model generates realistic responses but can inherit biases from dataset imbalances, overrepresenting common attacks like SQL injection and exposing the honeypot through repetitive patterns. To counter this, we ensured a balanced dataset with diverse attacks and legitimate interactions. Stochastic variability and post-processing further enhance response diversity improving engagement and deception against sophisticated attackers.

3.4 Deep Q-Network

Reinforcement Learning (RL) is employed in the honeypot system to optimize response selection, ensuring that interactions with adversaries appear realistic while maximizing engagement. This is achieved through a Deep Q-Network (DQN), which refines and selects the most appropriate response from a set of candidate responses generated by the Seq2Seq model.

Deep Q-Network was chosen due to its ability to handle high-dimensional state spaces [19] which are required while dealing with HTTP requests and responses. The HTTP requests and responses present a complex and vast state space due to the structured components like headers, payloads, and query parameters, often indicative of attack vectors such as SQL Injection. The neural network maps the high dimensional inputs into latent representations, capturing essential features only.

The Deep Q-Network (DQN) estimates Q-values, representing the expected cumulative reward for selecting an action (a) in a given state (s), accounting for both immediate rewards (r) and discounted future rewards (γ) [19]. The Q-value is calculated in equation (2):

$$Q(s,a) = E[r + \gamma_{(a)}, Q(s^{'}, a^{'})] \quad (2)$$

Actions with the highest Q-value are prioritized to maximize long-term rewards. For instance, in the interaction scenario depicted in Fig. 3, the Modify action exhibits the highest Q-value ($\sim$ *4000*), indicating its effectiveness in maintaining attacker engagement. Conversely, actions such as Allow, Block, and Terminate exhibit lower Q-values, reflecting their limited utility in prolonging interactions or capturing novel payloads. The Q-values are iteratively updated using the Bellman Equation [20] depicted in equation (3):

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma_{(a^{'})} \times Q(s^{'}, a^{'}) - Q(s,a)] \quad (3)$$

This approach incorporates immediate rewards (r) and future reward ($\gamma_{\max(a^{'})} Q(s^{'}, a^{'})$) from the subsequent state ($s^{'}$), enabling the agent to learn and adapt dynamically [26].

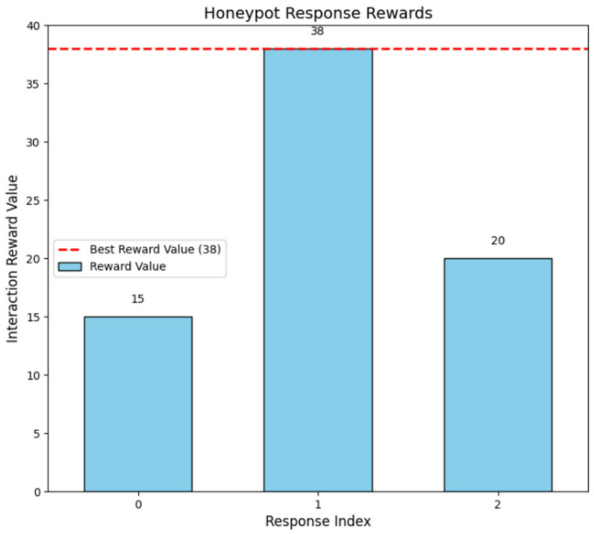

Fig.4: *Optimizing Honeypot Interactions with Response Rewards.*

The reward in this Deep Q-Network (DQN) context is a value generated by a custom reward function that serves as the core mechanism for optimizing the honeypot's interactive behaviour. Unlike traditional machine learning metrics, this reward is not a simple

accuracy score but an adaptive heuristic designed to prioritize specific security and intelligence-gathering goals. The strategy is two-fold: first, it provides a high positive reward for actions (responses) that contribute to prolonged engagement, specifically by increasing the Average Session Length (ASL), ensuring the attacker remains connected longer for observation. Second, it grants rewards for the discovery or elicitation of novel payloads or attack patterns, fulfilling the primary purpose of intelligence gathering. Crucially, this function incorporates an implicit penalization for responses that are too suspicious or unrealistic, maintaining stealth and realism. This dynamic, adaptive reward system aligns with the principles of reinforcement learning, where the goal is often to learn an optimal policy that maximizes long-term utility—in this case, maximizing intelligence collection while minimizing the risk of detection.

To formally represent this strategy, the custom reward function $R(s, a)$ is defined in equation (4):

$$R(s,a) = w_1 \cdot \Delta L + w_2 \cdot I_{novel} - w_3 \cdot P\_suspicion \quad (4)$$

where s denotes the current state of the interaction and a represents the agent's selected response. Here, ΔL represents the increment in session length (measured by the number of exchanges), rewarding the agent for sustaining engagement. I_{novel} is a binary indicator variable (0 or 1) that triggers when a previously unseen attack payload or pattern is elicited, fulfilling the intelligence-gathering objective. $P_{suspicion}$ is a penalty term applied if the generated response realism score (linked to the RRR metric) falls below a predefined threshold, ensuring the honeypot remains stealthy. w_1, w_2, and w_3 are weighting coefficients used to balance the trade-off between interaction length, intelligence capture, and system realism.

The neural network uses a multi-layer feed-forward architecture with 3 hidden layers and ReLU activation for efficient convergence. Key hyperparameters include a learning rate of 0.0001 for stability, a discount factor of 0.95 to prioritize long-term rewards, and a batch size of 64 for efficient gradient descent. The RL agent's reward function is designed to optimize interactions by rewarding prolonged engagement and the discovery of novel payloads, while penalizing suspicious responses that could alert attackers. This adaptive strategy balances learning, engagement, and stealth. As shown in Fig. 4, Response 1 achieves the highest reward (38), suggesting that it successfully prolonged interaction with the attacker while also introducing novel payloads.

This implies that the response was both engaging and deceptive enough to keep the attacker interacting with the honeypot. Response 2 (with a reward of 20) and Response 0 (with a reward of 15) are still valuable for learning, but they did not achieve the same level of engagement as Response 1. The decreasing reward values indicate that these responses either ended interactions sooner or did not introduce sufficient novelty to retain attacker interest.

To enhance adaptability, the learner module employs an epsilon-greedy policy, balancing exploration and exploitation to refine response selection. Initially, the model prioritizes exploration by selecting responses at random, thereby accumulating diverse attack interaction experiences. Over time, as learning progresses, the probability of selecting the optimal Q-value action increases, ensuring the execution of highly deceptive strategies. Additionally, a prioritized experience replay mechanism emphasizes rare and novel attack patterns, enabling the honeypot to more effectively adapt to emerging threats. This adaptive learning approach strengthens the system's resilience against sophisticated adversaries and continuously refines its interaction strategies to extract valuable intelligence from attackers.

3.5 Honeypot Instance

The honeypot workflow, as shown in Fig. 5, begins by capturing incoming HTTP requests, such as POST requests, and tokenizing them into fundamental components like request method, endpoint, headers, and payload. These tokens are then embedded as vectors and processed by a Seq2Seq Transformer model, which employs attention mechanisms to generate responses token by token.

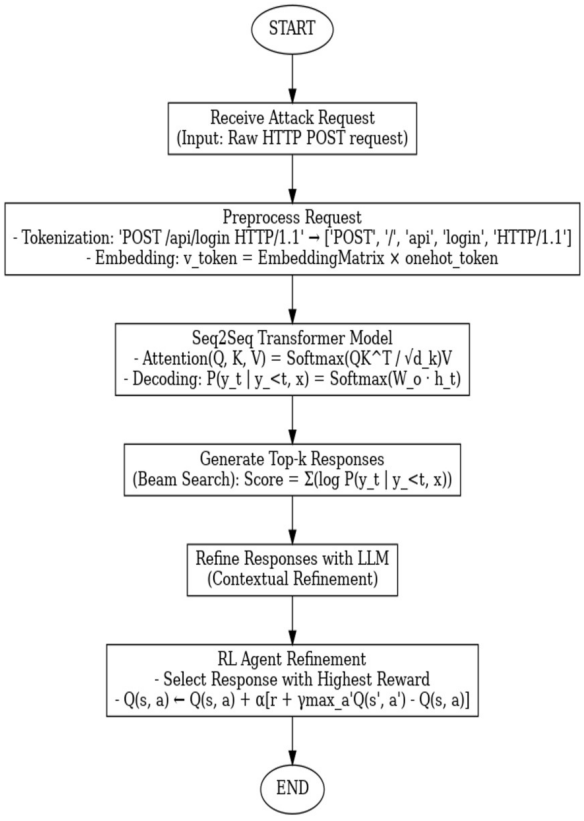

Fig.5: *Workflow of Honeypot Instance.*

To enhance response quality, beam search decoding selects the top-k responses based on probability scores, ensuring diversity and reducing. predictability. The generated responses are further refined using a Language Learning Model (LLM) to enhance realism and syntactic coherence, making them indistinguishable from legitimate system replies. Recent research highlights multi-step refinement strategies where an LLM first identifies problematic generations, then provides fine-grained critiques, and finally applies targeted refinements based on this feedback, resulting in responses that closely mimic legitimate outputs in both style and substance, demonstrating the LLM's effectiveness in producing highly realistic and coherent text [27]. A Deep Q-Learning (DQN) reinforcement learning agent then evaluates these responses using Q-values, prioritizing those that maximize attacker engagement while minimizing the risk of detection. The response with the highest Q-value is selected and sent back to the attacker, maintaining the illusion of an authentic system while subtly guiding interactions to extract valuable threat intelligence.

Table 3: *Formulae for Performance Evaluation.*

Evaluation Metric	Formula
RRR	(Server-like Responses ÷ Total Responses) × 100
SCA	(Contextually Appropriate Responses ÷ Total Responses) × 100
ORSR	(Engagement-Maximizing Responses ÷ Responses Evaluated by RL) × 100
ASL	(Total number of exchanges in all sessions ÷ Total number of sessions)

Table 4: *Training Configuration of the Seq2Seq Model.*

Parameter	Value
Learning Rate	0.0001
Batch Size	64
Epochs	10
Optimizer	Adam
Loss Function	Cross-Entropy

All interactions are logged for continuous model improvement, enabling the Seq2Seq Transformer and RL agent to refine their decision-making processes based on real attack patterns. The Seq2Seq model and the Deep Q-Network (DQN) agent operate within a sequential generate-and-select framework. Rather than integrating at the token level, the system follows a two-stage decision pipeline. In the first stage, upon receiving an attacker request, the Seq2Seq model generates a set of candidate responses using beam search with a beam width of five. These candidates vary in structure and semantic content, providing multiple plausible response options for the honeypot.

In the second stage, the DQN evaluates these candidates. The agent constructs a state representation that incorporates both the incoming request and the accumulated session context. The candidate responses produced by the Seq2Seq model are treated as the available actions. For each action, the DQN estimates a Q-value that reflects the expected long-term utility of selecting that response, with utility defined in terms of sustaining attacker engagement or eliciting richer payloads. The response associated with the highest predicted value is selected and delivered to the attacker. Through repeated interaction, this iterative cycle of generation, evaluation, response selection, and learning enables the honeypot to adapt its strategy and progressively improve its ability to maintain meaningful engagement with attackers.

4. PERFORMANCE EVALUATION

The performance of the proposed honeypot system was evaluated based on key metrics such as response realism, attacker engagement, and session length [21]. Comprehensive experiments were conducted to evaluate multiple aspects of the honeypot system, focusing on its ability to generate realistic responses, optimize interactions through reinforcement learning (RL), and sustain prolonged engagements with attackers. The RL module's effectiveness in selecting optimal responses was assessed by implementing a Deep Q-Network (DQN) to prioritize responses that maximize attacker engagement while minimizing the risk of honeypot detection. These metrics calculated as shown in Table 3, were measured in controlled environments simulating attacker scenarios interacting with the honeypot.

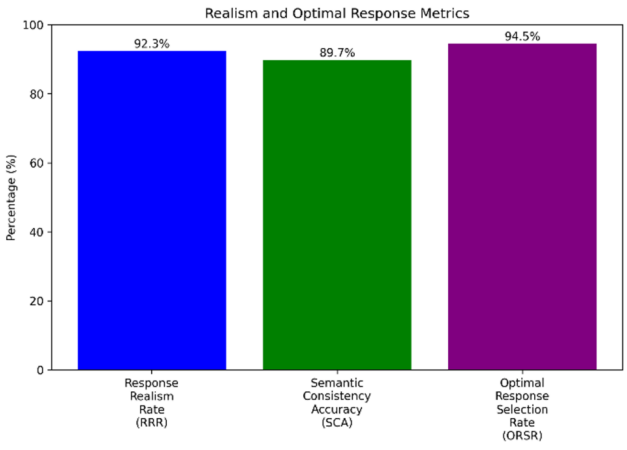

Fig.6: *Realism and optimal response metrics.*

These metrics rely on subjective assessment, and the human evaluation methodology for these metrics is described in the following subsections. These metrics align conceptually with the behavioural layers reflected in MITRE ATT&CK; RRR with procedural realism, SCA with technique-consistent semantics, and ORSR with tactic-aligned interaction shaping, which together support credible deception.

4.1 Experimental Setup

The honeypot was active for 60 days, with all experiments carried out on AWS EC2 t3.micro instances. The Seq2Seq model was implemented using PyTorch, with training performed on CUDA-enabled GPUs (CUDA version 12.4) to ensure efficient computation. The software environment utilized Python 3.9, PyTorch 1.13.1, MySQL 8.0 for dataset storage, and Selenium WebDriver for automated data collection. The dataset comprised *100,000* HTTP request-response pairs, gathered through automated scanning tools. This dataset included both attack types of SQL injection and command injection, along with legitimate interactions. The data was tokenized and normalized for consistency, and it was structured into two MySQL tables—requests and responses—linked via a unique response ID (res_id). The model training, as shown in Table 4, was conducted with a learning rate of *0.0001*, a batch size of *64, 10* epochs, a beam width of *5* for decoding, and the Adam optimizer.

4.2 Response Realism

The realism of generated responses was quantified using the Response Realism Rate (RRR) and Semantic Consistency Accuracy (SCA). Fig. 6 shows the RRR, calculated as the percentage of responses indistinguishable from legitimate server responses, achieved a score of *92.3%* based on human reviewer assessments. This high RRR highlights the honeypot's ability to produce convincing and authentic-looking responses. Similarly, the SCA, which measures semantic alignment between HTTP requests and responses, scored *89.7%*. These results confirm the Seq2Seq model's capability to capture contextual patterns in web interactions, enabling the generation of plausible and semantically accurate responses.

4.3 Response Selection Effectiveness

The RL-based response selection module was evaluated using the Optimal Response Selection Rate (ORSR), which measures the proportion of responses chosen by the RL agent that maximized engagement objectives. Fig. 6 shows the system achieved an ORSR of *94.5%*, reflecting its ability to prolong interactions and maintain plausibility. The RL module's reward function, designed to balance realism and interactivity, resulted in a Contextual Response Score (CRS) of *85.3* and an Average Response Reward (ARR) of *28.4*. These metrics validate the RL model's success.

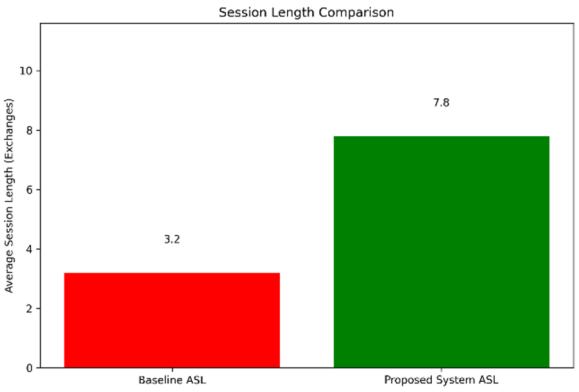

Fig.7: *Average Session Length Comparison.*

4.4 Attacker Engagement and Session Length

Attacker engagement was measured using the Average Session Length (ASL) depicted in Fig. 7, which increased from a baseline of 3.2 exchanges to 7.8 exchanges—a *143.5%* improvement—after integrating Seq2Seq and RL components. This underscores the honeypot's effectiveness in sustaining prolonged interactions by generating convincing replies. The baseline of 3.2 exchanges represents a standard honeypot setup without reinforcement learning or advanced machine learning capabilities, designed to handle simple attack attempts.

4.5 System Efficiency and Scalability

The system maintained an average response latency of 4 seconds due to beam search decoding, which remained acceptable for real-time interactions but slightly increased under high traffic. CPU usage stayed below 65% on AWS EC2 t3.micro instances, with swap memory helping during peak loads, highlighting the need for more RAM and optimizations. To handle high traffic, the honeypot scales by adding resources to instances and deploying across multiple availability zones. AWS auto-scaling ensures consistent performance and resilience, even during heavy traffic or sustained attacks.

4.6 Comparison with Existing Honeypots

This honeypot uses novel evaluation metrics like RRR, SCA, and ORSR making it effective now and a valuable model for future honeypots. Our intelligent honeypot addresses critical limitations of various existing high, low, and intelligent-interaction honeypots. Prior work such as SIPHON [22] and Honware [23] provides high-interaction honeypot capabilities with relatively low fingerprinting risk; however, both systems exhibit substantial limitations in scalability.

SIPHON, for example, depends on physical device forwarding, which introduces inherent hardware bottlenecks and restricts large-scale deployment. In contrast, the proposed Seq2Seq-driven honeypot operates efficiently on lightweight cloud instances (e.g., t3.micro), enabling rapid horizontal scaling without reliance on specialized hardware.

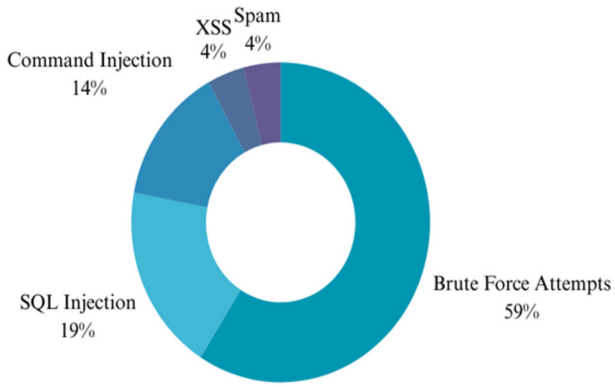

Fig.8: *Raw Requests captured by the Honeypot.*

IoTCandyJar [14] represents an early effort toward intelligent honeypots through the use of a Markov Decision Process (MDP). While effective in constrained settings, MDP-based methods typically encounter the curse of dimensionality when modeling high-complexity web traffic. The approach presented in this work mitigates these limitations by employing Deep Q-Learning, which is better suited for high-dimensional state representations derived from HTTP headers, payload structures, and session context.

In our experiments, a standard low-interaction honeypot (Baseline) sustained an Average Session Length (ASL) of only 3.2 exchanges. By comparison, our system achieved 7.8 exchanges, representing a 143.5% improvement over the standard baseline. This metric highlights the limitations of static scripts used in tools like Honeyd compared to our generative approach.

5. DISCUSSION

Deploying the intelligent honeypot on the public internet facilitated the collection of valuable data on early-stage exploitation attempts, particularly those targeting authentication mechanisms. The raw traffic was carefully analyzed and the distribution is shown in Fig. 8. The most prominent category of observed attacks was brute-force login attempts (59%), in an effort to gain unauthorized access. These attempts were typically high in volume and rapid in frequency, indicating widespread and automated credential stuffing campaigns.

In addition to brute-force activity, a key observation was that attackers frequently probed the login page using SQL injection (19%) and command injection (14%) techniques attempting to exploit vulnerabilities in user input fields. This aligns with existing research [24] indicating that authentication endpoints are among the most targeted entry points in web applications due to their potential for privilege escalation and unauthorized access. By actively responding to these probes with dynamically generated yet realistic responses, the honeypot effectively encouraged adversaries to engage in longer, more detailed attack sequences, providing richer insights into emerging tactics and payload variations. By actively responding to these probes with dynamically generated yet realistic responses, the honeypot effectively encouraged adversaries to engage in longer attack sequences, thereby tending to provide richer insights into emerging tactics and payload variations.

Unlike traditional low-interaction honeypots that merely simulate basic network services or return predefined responses, this system leveraged a deep learning-based response generation model to create a more interactive environment. This capability proved advantageous in eliciting multi-step attack strategies, as attackers often adjusted their inputs based on received responses. Such engagement provides critical intelligence, revealing patterns in automated attack scripts, behavioral tendencies of human adversaries, and variations in malicious payload construction. However, while high-profile web applications typically attract attackers more quickly than low-profile targets, their deployment requires careful consideration of network placement, integration with existing security infrastructure, and monitoring for evasion techniques [25]. Proper deployment ensures the honeypot remains an effective intelligence-gathering tool without becoming a liability that inadvertently exposes real systems to additional risks.

Despite its advantages, our intelligent honeypot is not entirely immune to fingerprinting attempts, a common challenge faced by deception-based security mechanisms [24]. Sophisticated attackers employ a range of techniques, such as analyzing response timing variations, subtle inconsistencies in HTTP headers, non-standard error messages, or unexpected deviations in application logic, to distinguish between genuine web services and honeypots. While our system integrates machine learning-driven response generation and reinforcement learning for adaptive decision-making, its effectiveness hinges on the model's ability to handle edge cases and adversarial probing techniques. Failure to do so may expose anomalies that can lead attackers to identify and bypass the honeypot, reducing its effectiveness in capturing meaningful threat intelligence.

6. ETHICAL CONSIDERATIONS

Honeypots pose risks, including misuse by attackers and unintended access by legitimate users. To prevent abuse and protect privacy, safeguards like ac-

cess controls and logging are in place. Sensitive data is encrypted or anonymized, and discovered website vulnerabilities are reported with remediation steps. Since this honeypot is for research, it operates in a controlled environment with limited interactions and privacy-focused data collection. These measures ensure meaningful cybersecurity research while minimizing risks.

7. CONCLUSION

This work introduces an intelligent honeypot for enhancing interactions with attackers targeting web applications. By integrating Seq2Seq models for realistic responses and reinforcement learning for engagement, the system maintains prolonged interactions and captures valuable attack data. Experiments showed a 92.3% Response Realism Rate, ensuring authentic replies, and a 143.5% increase in Average Session Length, proving its effectiveness in engaging attackers. Future work will explore large-scale deployments by evaluating the system across more powerful EC2 instance families, measuring throughput and latency under high-volume traffic, and assessing resilience through stress tests and controlled fault-injection scenarios. These studies will help determine how well the system generalizes beyond the constraints of t3.micro environments and behaves under real production pressure.

To improve robustness, the system will also be validated on datasets from different regions [28], time windows, and attacker profiles, alongside red-team and adversarial evaluations that test its resistance to adaptive or evasive techniques. Such experiments will reveal whether the observed performance gains persist across diverse threat landscapes.

Finally, the architecture can be extended beyond HTTP by introducing protocol-specific adapters and a shared encoder–decoder framework that supports FTP, SSH, JSON-RPC and other protocols. This modular design, combined with scalable containerized components and protocol-aware evaluation metrics, will enable efficient multi-protocol honeypot deployments while maintaining safety and performance guarantees.

AUTHOR CONTRIBUTIONS

Conceptualization, G.K.; methodology, A.V.; requirements analysis, A.V.; software, R.H. and R.B.; formal analysis, R.H.; data curation, A.C.; writing—review and editing, A.C.; visualization, R.B.; performance metrics, A.V., A.C., R.H., R.B.; supervision, G.K.; funding acquisition, G.K.

References

[1] D. S. Morozov, T. A. Vakaliuk, A. A. Yefimenko, T. M. Nikitchuk and R. O. Kolomiiets, "Honeypot and cyber deception as a tool for detecting cyber attacks on critical infrastructure," *3rd Edge Computing Workshop*pp. 81-96, 2023.

[2] J. L. Zobal, D. Kolář and R. Fujdiak, "Current State of Honeypots and Deception Strategies in Cybersecurity," *2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, Dublin, Ireland, pp. 1-9, 2019.

[3] L. Huang and Q. Zhu, "Adaptive honeypot engagement through reinforcement learning of semi-markov decision processes," *2019 10th International Conference on Decision and Game Theory for Security*, Stockholm, Sweden, pp. 196-216, 2019.

[4] Cobalt, "Cybersecurity statistics 2024," 2024. [Online]. Available: `https://www.cobalt.io/blog/cybersecurity-statistics-2024`.

[5] Varonis, "Cybersecurity statistics," n.d. [Online]. Available: `https://www.varonis.com/blog/cybersecurity-statistics`.

[6] L. Shi, Y. Li and M. Ma, "Latest Research Progess of Honeypot Technology," *Journal of Electronics Information Technology*, Beijing, vol. 41, pp. 498-508, Feburary 2019.

[7] I. Siniosoglou *et al.*, "NeuralPot: An Industrial Honeypot Implementation Based On Deep Neural Networks," *2020 IEEE Symposium on Computers and Communications (ISCC)*, Rennes, France, pp. 1-7, 2020.

[8] G. Betarte, Á. Pardo and R. Martínez, "Web Application Attacks Detection Using Machine Learning Techniques," *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Orlando, FL, USA, pp. 1065-1072, 2018.

[9] D. Grunova, V. Bakratsi, E. Vrochidou and G. A. Papakostas, "Machine Learning for Anomaly Detection in Industrial Environments," *Engineering Proceedings*, vol. 70, no. 1:25, 2024.

[10] S. Chaudhari, A. Thakur and A. Rajan, "Securing Digital Information Using Cryptography Techniques to Enhance IT Security," *Research Reports on Computer Science*, vol. 2, no. 3, pp. 13–22, 2023.

[11] A. Onukrane, H. K. Skrodelis, G. Merkurjeva and A. Romanovs, "Navigating Web Application Security: A Survey of Vulnerabilities and Detection Solutions," *2023 IEEE 64th International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS)*, Riga, Latvia, pp. 1-6, 2023.

[12] N. Provos, "Honeyd-a virtual honeypot daemon," *10th dfn-cert workshop*, hamburg, germany. vol. 2, 2003.

[13] Y. T. Abewa and S. Z. Melese, "Dynamic Interactive Honeypot for Web Application Security," *International Journal of Wireless and Microwave Technologies*, vol. 14, no. 6, pp. 1-14,

2024.

[14] T. Luo, Z. Xu, X. Jin, Y. Jia and X. Ouyang, "Iotcandyjar: Towards an intelligent-interaction honeypot for IoT devices," *Black Hat*, vol. 1, pp. 1-11. 2017.

[15] T. Yagi, N. Tanimoto, T. Hariu and M. Itoh, "Enhanced attack collection scheme on high-interaction web honeypots," *The IEEE symposium on Computers and Communications*, Riccione, Italy, pp. 81-86, 2010.

[16] S. Fatima1, S. Luqmaan and N. A. Rasheed, "Web scraping with python and selenium," *IOSR Journal of Computer Engineering (IOSR-JCE)* vol. 23, no. 3, pp.1-5, 2021.

[17] A. Vaswani *et al.*, "Attention is all you need," *31st Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, USA., 2017.

[18] M. Freitag and Y. Al-Onaizan, "Beam search strategies for neural machine translation," *arXiv preprint* arXiv:1702.01806, 2017.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[20] B. O'Donoghue, I. Osband, R. Munos and V. Mnih, "The uncertainty Bellman equation and exploration," in *Proceedings of the 35th International Conference on Machine Learning*, Stockholm, Sweden, 2018.

[21] P. Lanka, K. Gupta and C. Varol, "Intelligent Threat Detection—AI-Driven Analysis of Honeypot Data to Counter Cyber Threats," *Electronics*, vol. 13, no. 13, p.2465, 2024.

[22] J. D. Guarnizo *et al.*, "SIPHON: Towards Scalable High-Interaction Physical Honeypots," *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security*, pp. 57-68, 2017.

[23] A. Vetterl and R. Clayton, "Honware: A Virtual Honeypot Framework for Capturing CPE and IoT Zero Days," *2019 APWG Symposium on Electronic Crime Research (eCrime)*, Pittsburgh, PA, USA, pp. 1-13, 2019.

[24] M. Yamamoto, S. Kakei and S. Saito, "FirmPot: A Framework for Intelligent-Interaction Honeypots Using Firmware of IoT Devices," *2021 Ninth International Symposium on Computing and Networking Workshops (CANDARW)*, Matsue, Japan, pp. 405-411, 2021.

[25] K. Shivam1, M. Raval, N. Nrip and A. Sinha, "A research study on Web Application Security," *International Journal of Multidisciplinary Research Transactions*, vol. 4, no. 6, pp. 93-104, 2022.

[26] X. Glorot, A. Bordes and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS 2011)*, pp. 315–323, 2011.

[27] M. Wadhwa, X. Zhao, J. J. Li and G.Durrett, "Learning to Refine with Fine-Grained Natural Language Feedback," *arXiv preprint* arXiv:2407.02397, 2025.

[28] H. S. Annapurna and S. Devi, "Machine learning-based adaptive equalization with software-defined radio experimental setup," *ECTI Transactions on Computer and Information Technology (ECTI-CIT)*, vol. 19, no. 2, pp. 271–281, 2025.

Ananya Varadarajan is a Computer Science graduate specializing in cybersecurity. Her work focuses on security engineering in enterprise environments, particularly the design and optimization of SIEM platforms and automation-driven defense mechanisms. She is interested in applying artificial intelligence to strengthen security operations by improving threat detection, incident response, and overall operational efficiency. Her technical background includes cloud security, network defense, and the integration of machine learning techniques to advance automation within modern Security Operations Centers (SOCs).

Ashwin Chandrasekaran is currently pursuing his studies at PES University, Bengaluru. He is passionate about cybersecurity, with a strong interest in offensive security, including vulnerability assessment, penetration testing, and ethical hacking. His academic and practical work focuses on understanding real-world attack techniques and developing innovative security solutions to strengthen modern web applications and systems.

Rachana Binumohan is an Associate Consultant who earned her B.Tech in Computer Science and Engineering with a specialization in Artificial Intelligence and Machine Learning from PES University. Her research interests focus on reinforcement learning and adaptive intelligent systems, including sequence-to-sequence models integrated with reinforcement learning for dynamic decision-making. She has also worked on machine learning–based pest detection systems in agriculture, applying AI techniques to solve real-world challenges through predictive and data-driven approaches.

Rahul Huliyar Ravishankar is a Software Developer who earned his B.Tech in Computer Science and Engineering with a specialization in Artificial Intelligence and Machine Learning from PES University. His research interests include machine learning, deep learning, reinforcement learning, and intelligent systems, with a focus on adaptive models such as sequence-to-sequence architectures integrated with reinforcement learning. He has experience building scalable applications, designing web APIs, and deploying AI-driven solutions that bridge advanced machine learning with real-world software engineering.

Gokul Kannan Sadasivam is a Professor of Computer Science and Engineering at PES University and has strong technical foundation in Network Security and Machine Learning, acquired through a double master's degree and thirteen years of work experience in teaching, research, and industrial roles. Research skills/interests in intrusion detection systems, network forensics, machine learning, and data mining. Published several papers in reputed conferences and journals. Good team player with excellent communication skills. Highly motivated to learn new enhancements in technology and a flair for creativity.