AI-Driven Honeypot: An Innovative Approach to Adaptive Cyber Security Defense

ARTICLE HISTORY
Received 6 January 2026
Accepted 27 March 2026
Published 7 July 2026

Danny Corbett
Sheffield Hallam University
Cyber Security
Sheffield, UK
dannyycorbett@gmail.com
ORCID: 0009-0007-1651-075X

Shahrzad Zargari
Sheffield Hallam University
Cyber Security
Sheffield, UK
s.zargari@hallam.shu.ac.uk
ORCID: 0000-0001-6511-7646

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2026

D Corbett, S Zargari, "AI-Driven Honeypot: An Innovative Approach to Adaptive Cyber Security Defense", Latin-American Journal of Computing (LAJC), vol. 13, no. 2, 2026.

AI-Driven Honeypot: An Innovative Approach to Adaptive Cyber Security Defense

Honeypot Impulsado por IA: Un Enfoque Innovador para la Defensa Adaptativa de la Ciberseguridad

Danny Corbett
Sheffield Hallam University
Cyber Security
Sheffield, UK
dannyycorbett@gmail.com

Shahrzad Zargari
Sheffield Hallam University
Cyber Security
Sheffield, UK
s.zargari@hallam.shu.ac.uk

***Abstract*—** As cyber threats continue to grow in sophistication, the need for intelligent and adaptive defense mechanisms becomes increasingly more critical. This research investigates the integration of Artificial Intelligence (AI) into a honeypot system to distract, mislead through deception, and engage potential cyber attackers. The primary research question to answer was: “How can AI-driven adaptive deception improve the effectiveness of honeypots in cybersecurity?” To address this, a high-interaction honeypot was developed on a HTML website to be perceived as a reverse shell, with the implementation of OpenAI’s GPT-4o model to respond, impersonating a Linux terminal, while silently tracking and logging the attacker, and classifying all commands into three sub-categories – Safe, Suspicious and Malicious. The core methods included command logging, AI-driven risk classification, dynamic fake filesystem manipulation, and the escalation of behavior based on the attacker's actions. Attack simulations were performed by highly credible third-party cybersecurity experts to evaluate the honeypots effectiveness in engaging and tracking the attacker for as long as possible. The findings suggest that AI integration significantly improved the realism and engagement level of the honeypot, both in terms of enhancing intelligence gathering and the improvements from traditional static honeypots. However, full automation of behavioral escalation tuning remains an area to further explore. Overall, this study demonstrates that the integration of AI within traditional honeypot strategies can significantly enhance cyber defense systems.

***Keywords*—** *honeypot, artificial intelligence, cybersecurity, adaptive deception, GPT-4o, intrusion detection*

***Resumen*—** A medida que las ciberamenazas se vuelven cada vez más sofisticadas, la necesidad de mecanismos de defensa inteligentes y adaptativos se vuelve cada vez más crítica. Este proyecto investiga la integración de la Inteligencia Artificial (IA) en un sistema honeypot para distraer, engañar y atraer a posibles ciberatacantes. La principal pregunta de investigación fue: "¿Cómo puede el engaño adaptativo basado en IA mejorar la eficacia de los honeypots en ciberseguridad?". Para abordar esto, se desarrolló un honeypot de alta interacción en un sitio web HTML para que se percibiera como un shell inverso. Se implementó el modelo GPT-4o de OpenAI para responder, y suplantar una terminal Linux, mientras rastreaba y registraba silenciosamente al atacante y clasificaba todos los comandos en tres subcategorías: seguro, sospechoso y malicioso. Los métodos principales incluyeron el registro de comandos, la clasificación de riesgos basada en IA, la manipulación dinámica de sistemas de archivos falsos y la escalada del comportamiento en función de las acciones del atacante. Expertos externos en ciberseguridad de alta credibilidad realizaron simulaciones de ataques para evaluar la eficacia de los honeypots a la hora de interactuar y rastrear al atacante durante el mayor tiempo posible. Los resultados sugieren que la integración de la IA mejoró significativamente el realismo y el nivel de interacción del honeypot, tanto en términos de mejora de la recopilación de inteligencia como en comparación con los honeypots estáticos tradicionales. Sin embargo, la automatización completa del ajuste de la escalada del comportamiento sigue siendo un área que requiere mayor exploración. En general, este estudio demuestra que la integración de la IA en las estrategias tradicionales de honeypots puede mejorar significativamente los sistemas de ciberdefensa.

***Keywords*—** *honeypot, inteligencia artificial, ciberseguridad, engaño adaptativo, GPT-4o, detección de intrusiones*

I. Introduction

The cybersecurity landscape is an ever-evolving environment, with attacks on systems rapidly becoming more sophisticated and advanced. To counter the development of malicious threats, cybersecurity is required to constantly evolve and adapt, to stay ahead of emerging risks. A successful tested example of achieving this is the honeypot concept. A honeypot is defined as an information system or system resource, designed to serve as an attractive target for attacks [1]. The purpose of a honeypot is to detect, analyze and distract cyber attackers from gaining unauthorized access to the real system [30]. This determines a honeypot being an incredibly important resource to implement, as it can collect data to analyze attackers' tactics through their interaction with the honeypot, gaining insights into attack patterns, tools and techniques used, to help improve and develop the systems’ defense strategies. A honeypot will also slow down the attacker’s progress through diversion and deception, waste their time and efforts, frustrate the attacker, and reduce the risk of attacks on the real system’s infrastructure.

Within cybersecurity, deception has historically been applied from the earliest honeypots, such as Fred Cohen’s Deception Toolkit in 1998, which provided fake services as a decoy machine [2]. Over time, honeypots advanced to become more sophisticated, like Honeyd, which simulated networks Simultaneously, although cyberattacks have become more advanced, concerns have also emerged about the uprising limitations of static honeypots. However, the implementation of machine learning (ML) and artificial intelligence (AI) into honeypots is proving to be the next evolution to combat the complexity of new and advanced cyber threats [3].

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2026
DOI: https://doi.org/10.33333/lajc.vol13n2.01

Despite the growing application of AI in cybersecurity, combining AI with Reinforcement Learning (RL) is relatively unexplored. Research from [4] suggested RL as a machine learning paradigm which identifies the optimal policy based on an action, shown by an action (a_t) in state (s_t), leading to the optimal policy while maximizing the reward (R (s_t, a_t)). After a training process, the RL model interacts until finding the optimal policy. Furthermore, RL has proven to be very efficient, evidenced by a survey from [5] that compared RL algorithms, including AdaBoost. They concluded that the AdaBoost algorithm achieved a low false positive rate (between 2.7% - 3.5%) and a high detection rate (between 90% - 99.3%). While AdaBoost is not used in this research, the study highlights the suitability of ML for detecting and classifying commands and attacks, which we explore later in this paper.

Conversely, static honeypots are defined as honeypots with predefined, unadaptable behavioral responses, regardless of the attacker's interactions. According to [6], these types of honeypots can be used for either detection or deception, which can maximize their effectiveness within the Japonica framework [32]. However, these static honeypots can be easily revealed by attackers through fingerprinting, which is the process of identifying unique characteristics of the honeypot, or through probing techniques to expose them. In contrast, an AI-enhanced honeypot; particularly, those that utilize RL, can adapt with dynamic responses from real-time analysis of an attacker's behavior. This is a gap to explore in the development of cyber security defenses for adapting to threats more effectively.

Conversely, honeypots can be classified into low-interaction and high-interaction systems. Low-interaction honeypots are limited to static services or responses, which are easily fingerprinted. Meanwhile, high interaction systems provide a realistic environment that allows an attacker to interact with a real operating system or an emulated one. Recent work [7] describes high interaction honeypots as being capable of obtaining rich data, gaining insights into attacker behavior and tactics. The evolvement of honeypots is crucial to modern threat intelligence, as low-interaction honeypots are much less efficient and stealthy than adaptive, intelligent high interactive honeypots, creating a need for enhanced honeypot systems. Nevertheless, the risk posed by high-interactive honeypots introduces massive security risks [7] to the actual network environment; e.g., the exposure of a real shell. To address this risk, a honeypot was deployed on port 80 to appear as a web interface, posing no risk associated with Secure Socket Shell (SSH) connections, while still appearing as a legitimate system terminal.

This research aims to develop an AI-enhanced honeypot that detects, analyses, and adapts using a heuristic deception policy in conjunction with the AI model to become as stealthy, realistic and efficient as possible. The objectives of this honeypot include the implementation of Cowrie on Port 22 to provide another version of honeypot to the intended main web interface honeypot. By this means, our aim is using AI to respond to attackers by recording, classifying, and analyzing all data while adapting its deception strategies. Our research also explores how a honeypot can easily escape being fingerprinted through adaptation – by modifying its behavior, based on an attacker interaction to make the deception more effective.

The research will be guided by four key questions that need to be addressed:

- Can the integration of Artificial Intelligence effectively deceive attackers through the ability to become adaptable and realistic?
- Can an AI-controlled honeypot collect higher quality data when compared to a static honeypot?
- Does AI-enhancement identify and classify commands effectively?
- How effective is an adaptive, heuristic-driven deception module in responding more intelligently to attacker behavior over time, to prolong attacker engagement and improve attacker deception?

Our experiments follow the network set-up shown in Figure 1.

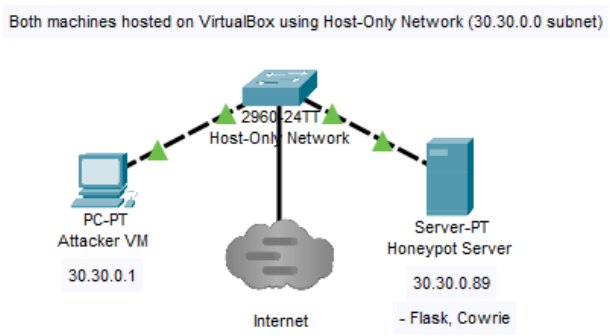

Fig. 1. AI-driven honeypot intended framework

The proposed scope includes AI classification, an SSH honeypot, reverse shell simulation with AI-generated responses, and attacker logging. However, there are also some limitations, including that our honeypot is constrained within a controlled network, emulating attacks instead of testing the honeypot using real advanced persistent threats (APTs) from external sources. Despite this, we contribute to the rapidly growing cybersecurity field in intelligence defense strategies by demonstrating how AI can offer insights with adaption, deception, data collection and classification.

II. LITERATURE REVIEW

The purpose of this literature review is to understand the current reliable research relating to the efficiency of AI-controlled honeypots, comparing them with traditional honeypots, as well as aiming to understand different strategies, implementation, and findings about AI driven honeypots. By this means, we can understand the current state-of-the-art completed within this sector, growing trends, and potential gaps where this research can focus on. As the use of AI in honeypots is relatively new, the literature review will focus on recent publications, from the initial frameworks to the more recent use of adaptive intelligence and real-time response.

A. Honeypot Framework

Honeypots have gained a reputation as a valuable tool to attract, deceive, and analyze attackers. However, their effectiveness is decreasing due to advancements in fingerprinting techniques, which can identify a honeypot.

Research presented in [8] proposed a comprehensive study on honeypot fingerprinting in 'Gotta Catch 'em All', using a multi-stage framework. The multistage framework evaluates fingerprinting on honeypots across the network layers of the OSI model and assesses typical honeypot implementations and detects inconsistencies within protocols, network responses, and delays to reveal the honeypots as what they are. The proposed solutions to the exposed indications of honeypots like Cowrie or Dionaea consist of protocol obfuscation, which alters network responses to mimic real systems, and behavior randomization, that introduces variability in responses, interactions and delays for mitigating fingerprinting. The research shows the importance of adaptability and realism within honeypot deployment to keep up with the evolving use of malicious techniques. However, AI or adaptive deception is not incorporated in this particular research, signaling an opportunity to further explore and improve using intelligent models.

B. Use of Large Language Models (LLM) in Deception Systems

Recent work in LLMs have driven a new wave of high-interaction honeypots. In [9], it is concluded that by fine-tuning an open source LLM with data from attacker commands, a honeypot can effectively generate realistic AI responses, demonstrating the potential of LLMs to improve threat detection and analysis. Their results show promise in enhancing realism and engagement within honeypots. Although the LLM may be vulnerable to fingerprinting, a potential research gap is devised for synthesizing and combining [8] fingerprint evasion strategies.

Similarly, research done in [10] investigated GPT-3.5 in an SSH-based environment. The method consisted of analyzing 1,400 pairs <request, response> across three datasets using GPT-3.5, finding that, while it maintained context within outputs, it struggled with long-session coherence and realism. After adapting a paraphrase-mining approach, the study achieved a macro F1 score of 77.85%, which was used to evaluate the performance of the model in terms of precision and recall. The higher the percentage, the more convincing LLM generated response, leaving a valuable potential gap to improve.

In contrast, research in [11] found that the implementation of LLMs is capable of more realistic interactions, by producing a LLM (LLMPot) that effectively emulates ICS protocols. ICS networks are vulnerable to cyber-attacks due to their ease of connectivity. Therefore, the LLMPot was introduced to implement dynamic protocol emulation in real-time. The results suggest that while LLMs were challenged by SSH environments, they were very effective for more structured, protocol-based honeypots like ICS. Unlike, the generalized interaction model proposed in [9], LLMPot's interactive approach concluded that context-specific honeypots benefit from specialized models, while multi-vector, adaptive honeypots benefit from generalized intelligence.

In addition to this research, in [12] a honeypot called 'DecoyPot' is featured. It simulates API interactions in web environments using LLMs. The system proved to be highly engaging, demonstrating potential in how generative models can be highly functional in a HTTP-based service. It also justifies implementing a web-interfaced fake reverse shell to showcase how effective LLMs can be within highly interactive honeypots.

The literature review has also shown that each approach to evaluating LLM-driven honeypots can be thought of as different frameworks in different ways. While [9] and [11] both prioritize improving the realism in honeypots, they differ between flexibility and being domain specific. In contrast, [10] focused on establishing a standardized methodology, ensuring both scalability and replication.

C. Adaptive Intelligence and Real-Time Response

Beyond LLMs, broader AI-based honeypots have gained interest in the last year. Research in [13] found that traditional honeypots, that rely on static configurations, are becoming less effective against advanced and evolving cyberattacks. Their AI-driven model collected over 100 GB of data in 24 hours, maintaining attacker interaction for 40% longer than static honeypots, with a 90% detection rate, compared to the 65% rate for static honeypots. These results reinforce the critical effectiveness of adaptive honeypots, from defending against a range of zero-day exploits, advanced persistent threats (APTs) and polymorphic malware. The study split the model framework into different layers:

- External attackers, using real cyber-attacks. Unlike our proposal which intends to use simulated attacks to test it locally.
- Honeypot interaction environment in which the AI makes responses based on the attack, comparable to our proposed honeypot)
- Data collection layer to capture network traffic, interactions and attack data, which is crucial to recording results.
- Security analysist for threat intelligence gathering from data collected in the previous layer, and
- AI-based adaptation engine where an AI processes data to adapt the honeypot behavior in real-time.

This blueprint can be used to effectively to simulate an AI-driven adaptive honeypot that is more efficient that a static honeypot.

Like [13], research from [14] used AI-enhanced honeypots to address zero-day exploits. Traditional honeypots are often not dynamic enough to challenge these types of exploits. However, the AI-controlled honeypot was successful in predicting exploit attempts in real-time. The AI-enhanced honeypot achieved a 92% detection rate for zero-day exploits, while traditional honeypots got 75%, highlighting the significant improvement these types of honeypots can achieve. Despite the advantages, the AI algorithm found a high number of false negatives, which negatively impacts the integrity and reliability of the AI. This study outlines the clear difference between the two honeypots and highlights the importance of honeypots being adaptable using AI. It provides valuable insights from which any future research can be built on; especially, the importance of recognizing how false negatives could become a problem.

In addition to external threats, internal threats also require real-time cybersecurity precautions. Research in [15] addressed how sophisticated internal threats need real-time, efficient cyber security measures in place. The study uses real SSH honeypot logs mapped to the MITRE ATT&CK

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2026
DOI: https://doi.org/10.33333/lajc.vol13n2.01

framework, leveraging Retrieval-Augmented Generation (RAG) and K-Means clustering to classify attacker behavior. The advantage of this research lies in the use of high-interaction honeypots combined with an AI-driven approach to automate data analysis. This enables threats to be detected as quickly as possible, leading to better incident responses to mitigate threats, and enhance anonymity detection. However, the disadvantage of this approach is that it only detects a specific type of attack.

D. Implementation of ChatGPT within honeypots

Other studies have explored the use of generative AI, such as ChatGPT (model GPT-3.5) within honeypots. In an educational seminar [16], it was demonstrated how an AI-controlled honeypot can mimic a Linux server. The seminar also showed how, after a potential attacker puts a malicious script in the HTTP server to create a reverse shell, the attackers begins unknowingly to interact with ChatGPT, distracting and misleading them. However, their system could be easily broken using prompt injections, and could not handle file uploads or network requests, highlighting a potential gap to explore and solve.

Further evaluation of ChatGPT's potential in honeypots was conducted in [17]. Here, Elasticsearch and SSH honeypot logs were mapped to the MITRE ATT&CK framework. In two weeks, 627 Elasticsearch requests and 73 SSH attack sequences were examined, finding that while the MITRE ATT&CK Mapping accuracy was 72.46% for Elasticsearch and 98.84% SSH accuracy, ChatGPT achieved 96.65% Elasticsearch and 97.26% SSH accuracy. This proves to be crucially important in Elasticsearch logging, with little difference in SSH logging. In terms of obfuscation detection, ChatGPT did identify the obfuscation well, but it also produced a high number of false positives (30.46% of request bodies and 7.5% of targeted URIs falsely recorded). These findings are similar to those in [14], supporting the issue around AI-driven honeypots recording false negatives. To conclude, ChatGPT was effective for automated honeypot analysis, but had a big setback with the high false positive rate, allowing for future work to investigate their role in incident response. The false positive setback proves to be a big weakness in AI-driven honeypots across multiple studies and is something to consider for future research.

E. Synthesis of Findings and Research Gaps

In the previous section, different findings and problems have been highlighted, which can be later explored in the methodology. Our proposal aims to design a resilient honeypot against prompt injection and enhance the other weaknesses. These findings and issues consist of the following:

- The use of AI may result in a high number of false positives.
- How prompt injections can easily detect the use of AI instead of a real system.
- How a high-interaction honeypot can keep attackers engaged.
- How AI can detect threats to log and report, in a rapid, efficient way.
- The most efficient framework for a high-interaction honeypot to be as realistic as possible.

Lastly, the literature review identified what makes an effective AI-driven honeypot and it is critical that these learnings are considered during the design phase:

- Adaptability: Research in [13] and [14] showed that adaptability using AI models significantly improves performance metrics.
- Balancing realism and security: Research in [9] created a high-interactive and realistic honeypot, but this came with the flaw of being vulnerable to prompt injection and fingerprinting. While research in [8] explored and concluded that fingerprinting cannot be completely mitigated against.
- AI in Detection and Analysis: Research showed through RAG and K-Means clustering [15] as well as ChatGPT classification [17] that AI can effectively identify, classify and report attacks despite accuracy problems.

III. Methodology

A. Introduction

This section will cover the methodology, outlining the tools and techniques used to create the honeypot, and the data it produced. We aim to fulfil the gaps highlighted in the literature review, creating an AI-controlled honeypot that responds to attackers in the most realistic method possible, without being broken by prompt injection, as well as recording the data as efficiently as possible, without frequent false positives.

This methodology was also guided by the four key questions that needed to be addressed, highlighted in the introduction.

B. Research Design and Approach

Figure 2 illustrates the structured design of the honeypot to demonstrate how an attacker would interact with it.

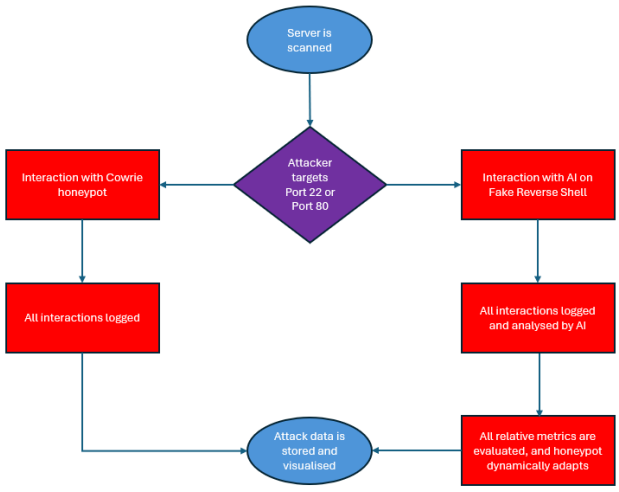

Fig. 2. Framework of Methodology

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2026
DOI: https://doi.org/10.33333/lajc.vol13n2.01

1) Honeypot Design

An experimental design was adapted to implement AI and enhance the honeypot system. This approach consisted of:

- Setting up an Ubuntu Linux Server and an Ubuntu Linux Desktop on the same network.
- The creation of a HTTP web page
- The implementation of Cowrie on Port 22
- Developing a web-based fake reverse shell to capture attackers
- Implementation of GPT-4o to act like a Linux server and adapt to the threats.

The GPT-4o model was chosen for being more intelligent within its responses than the GPT models from the literature review (GPT-3.5). Such models were also prone to prompt injection, and recording high false positives, which are critical limitations [14], [16].

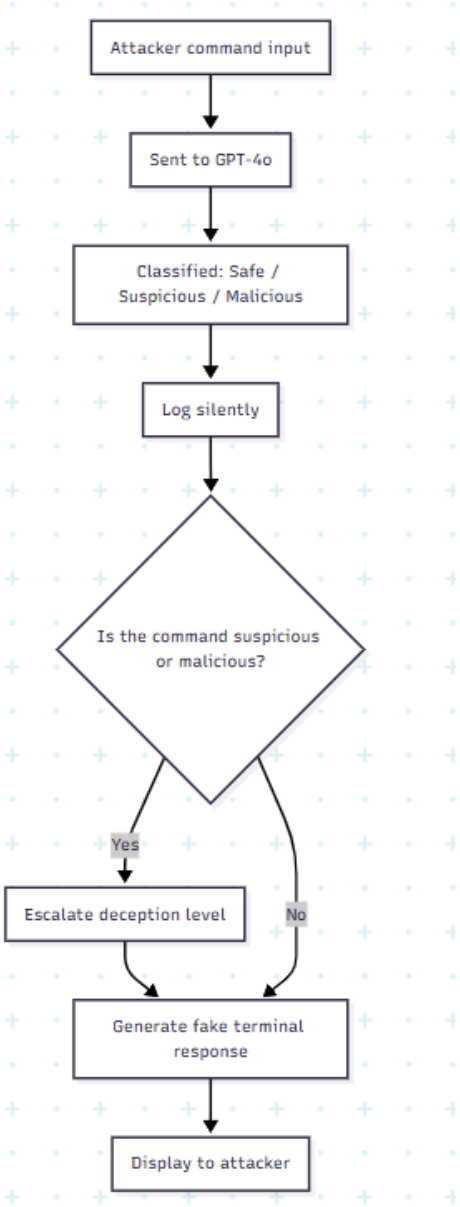

Fig. 3. AI-Driven Honeypot Flowchart

2) Honeypot Approach

The GPT-4o model was implemented in three separate sections to provide three different functions: (i) To respond to commands, (ii) to classify commands, and (iii) to generate fake files. Each model would require unique training to its unique task.

- The collection of the data that the honeypot gathered can be monitored and analyzed in real-time, including the GPT-4o model, which identifies the risk level of commands entered to the categories: Safe, Suspicious or Malicious.
- Allowing the attacker some privileges, such as the creation and deletion of both directories and files, to keep the attacker engaged.
- Development of the honeypot to adapt, based on interactions with it.

Figure 3 displays the framework about the honeypot operation: From when a command is entered to the output it responds with. The most crucial and key issue that was identified from the literature review for our honeypot to function efficiently was making sure the prompt injection could not be broken.

While researched studies used GPT-3.5, we implemented GPT-4o to mitigate against prompt injection that causes the AI to break character. In [17] an examination of different models' abilities was conducted to understand and generate language; particularly in complex scenarios. Here, GPT-3.5 scored in the bottom 10%, GPT-4o scored in the top 10%, highlighting the difference and need for a smarter model to be implemented. As such, GPT-4.0 has been used in the design.

C. Tools and Techniques

1) Network Configuration

In a controlled environment that was created for our proposed honeypot, a Linux Ubuntu Server and a Linux Ubuntu Desktop were created on the same internal network, using Oracle VirtualBox. The honeypot was deployed onto the server and configured such that, after an Nmap scan, it displays ports 22 and 80 as open, so that an attacker believes they have a couple of entry points to the server.

2) SSH Honeypot Deployment

On Port 22, Cowrie [29] was downloaded, which is a virtual SSH-based honeypot, designed to log brute force attacks. Cowrie simulates a SSH login system, emulating a session that will never grant SSH access. The objective is to observe the attacker by logging behavior and passwords entered on a simulated UNIX system [19]. Another advantage of Cowrie is that it can generate structured, detailed logs to gain meaningful insight into attacker behavior and tactics. In fact, a study [20] reports that it achieved an f1-score of 89.8%, proving Cowrie's effectiveness in collecting highly reliable data. In [19], the effectiveness of Cowrie relies on its Support Vector Machine classification accuracy, which scored 97.39%, highlighting Cowrie's role in AI-enhanced intrusion detection.

3) Web-Based Reverse Shell on Port 80

In addition, on Port 80, a Flask-Based Web Server was set up using Python, for the web-based fake reverse shell to be established there. This Fake Reverse Shell was the main

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2026
DOI: https://doi.org/10.33333/lajc.vol13n2.01

component of our proposal since an attacker is tricked into interacting with the AI-controlled honeypot, which uses GPT-4.0 to simulate exactly what a real Linux server would look and respond like in order to adapt to the attackers' movements. All commands and AI-generated responses are logged and analyzed by an AI model. The code was completed primarily using Python, due to:

- Extensive Libraries Supported, including web development frameworks like Flask, which was used because of its ease to deploy and integrate both the web-based interface and adaptive AI [21], as well as mimicking real world web exploits.
- Ease of applying AI and machine learning integration.
- Prototyping – ease of debugging [22].

The code consists of the creation of an HTML website to appear as a reverse shell on the servers IP address, with AI implemented to respond to all commands entered into it to simulate the real server, as well as AI to log and determine the risk level of each command.

The code also allowed for certain escalated privileges, such as creating and deleting both files and directories – in order to keep attackers engaged by misleading them into thinking they have some privileges, and therefore, deceive them into thinking privilege escalation on the server may be possible. The code also enabled the AI to generate fake files based off the filename.

The honeypot is displayed as shown in Figure 4 – The terminal HTML appears as a reverse shell as the user 'Dave' to manipulate attackers.

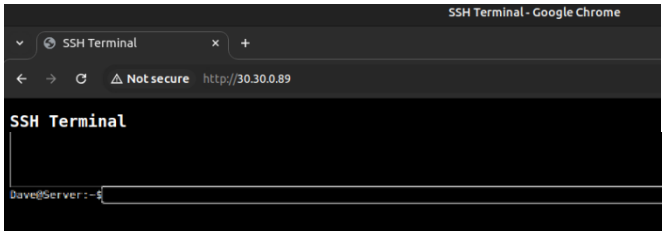

Fig. 4. The AI-controlled Web-based reverse shell

4) AI-driven Adaptive Deception

Initially, a static honeypot was deployed with predefined responses to test connections, while a basic keyword-matching method and a 'suspicious threshold', which would implement after the quantity of commands increased, was used to detect suspicious command patterns. Once this was working correctly, a rule-based adaptive policy, driven by command-classification thresholds and LLM-generated responses, was developed to be able to dynamically adjust responses, based on the attacker's interaction. Both suspicious and malicious command types are flagged and categorized by the GPT-4o AI model. To assess the effectiveness of the adaptive deception model, it was tested in comparison to the original static honeypot. The metrics included the quantity of suspicious and safe commands, risk level success rate, and attacker session duration. This test concluded that the AI's decision making was much more advanced and reliable, increasing the validity and therefore the performance of the honeypot. The suspicious threshold was maintained, as it tracks the user via the client IP count and flags them once they exceed the threshold, alerting the IP address as a likely attacker.

The adaptive module's main focus, however, was to generate responses that the honeypot would output, instead of using Flask's application that used static, rule-based, pre-defined responses. The adaptive model was built to learn from attacker behavior in real time so it can adapt and dynamically generate outputs based on attacker behavior. This can then keep the attacker engaged for the maximum time as well as track the attackers' steps and behavior, proving to be much more efficient. The adaptability of the honeypot was implemented through tracking the attacker's state, such as the IP addresses deception level, suspicious threshold, and session, and changing over these variables.

An example of the AI-driven adaptive deception implemented was to monitor behavior from a suspected IP address trying to open files, which then triggered password prompts, which in turn would further escalate the honeypot's hints and 'potential weaknesses' to make the attacker believe that they are getting somewhere with the attack. The attacker would then be consistently finding what they believe to be new information, without realizing it is all generated information. This is crucial in keeping the attacker engaged in the honeypot for as long as possible.

5) Tools and Techniques to simulate attacks on the honeypot

- Nmap was originally used to perform reconnaissance and gather information on open ports.
- Attempts to SSH into the server or create a reverse shell on the client machine.
- Linux commands such as 'ls', 'cd' 'whoami' which are considered as non-malicious, to navigate the system, and determine if the AI classifies these commands as usual behavior.
- Malicious Linux commands such as 'wget', 'reboot', 'rm' to verify the honeypot is classifying the IP as an attacker, and that the honeypot adapts to the specific command, responding in a realistic manner.

D. Data Collection and Analysis

All activity and interactions were logged in real-time through Flask's logging methods. This data can be accessed in the server but was also mapped to a comma delimited text (CSV) file.

The developed software classifies the following data elements:

TABLE I. DATA THAT THE HONEYPOT LOGS

Element	Example of Output
ID	1
Timestamp	2025-03-12 02:22:03
IP Address	30.30.0.1
Command	whoami
Risk Level	Suspicious
Current Directory	home
Response	Dave

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2026
DOI: https://doi.org/10.33333/lajc.vol13n2.01

The risk level was determined by the GPT-4o AI model, as instructed to define what level of risk each command should be classified as, between Safe, Suspicious, and Malicious.

All SSH login attempts on port 22 were logged via Cowrie logs.

E. Ethical Considerations

1) Lab set-up

To ensure ethical compliance, the honeypot was deployed in a controlled lab environment that was set up just for this purpose. This prevents honeypots from being accessed over the internet or any internal network, avoiding unethical cybersecurity practices and limiting external factors. The collection of data adhered to privacy and legal standards, without the use of personally identifiable information.

2) AI Jailbreaking

Reference [23] states that AI jailbreaking refers to a technique in manipulating AI models to bypass restrictions, causing the system to execute malicious instructions which may violate relative policies and produce harmful outputs. It was noted that this research could be subject to attackers manipulating the AI-controlled honeypot to leak sensitive information or create harmful content.

However, [18] does suggest that while AI jailbreaks are a big problem for GPT models, they found that the model GPT-4o has a considerably lower amount of incorrect behavior rate on disallowed or sensitive content, as illustrated in Figure 5.

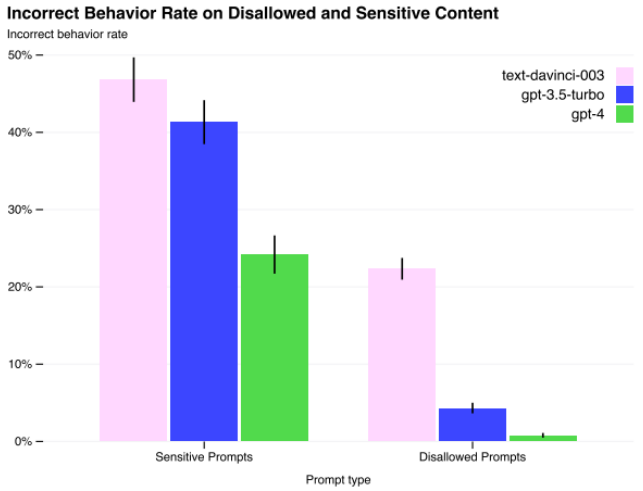

Fig. 5. Rate of incorrect behavior on sensitive and disallowed prompts. Lower values are better GPT-4 RLHF has much lower incorrect behavior rate compared to prior models

Therefore, through using the more advanced GPT-4o model instead of the 3.5 model, as well as flagging potential attempts to manipulate the AI, can mitigate against the threat of AI jailbreaks.

IV. Results and Discussion

This section will present data that the honeypot produced through logs, adaption from attacker behavior, AI decisions, and the effectiveness of the honeypot at appearing realistic and deceptive. Results will be generated from CSV logs using SQLite, Cowrie and Flask, to be further critically analyzed and discussed in this section.

Upon initial discovery of the servers IP address, an Nmap scan shows port 22 and port 80 as open, both with the honeypots deployed, Figure 6.

```
danny@danny-VirtualBox:~/Desktop$ nmap -p 22,80 30.30.0.89
Starting Nmap 7.80 ( https://nmap.org ) at 2025-04-23 15:49 BST
Nmap scan report for 30.30.0.89
Host is up (0.013s latency).

PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 1.05 seconds
```

Fig. 6. Results from Nmap scan of the honeypot server

A. Data Overview

The AI-driven honeypot was found to be incredibly successful because it never broke character in all the tests that were undertaken. After multiple different input commands to attempt to get the AI to break character, the responses remained true to a Linux server. This testing was completed because of the literature review findings from [16], which had shown significant issues breaking character within their honeypot design. This is a notable advantage of this honeypot, as the attacker is less able to obtain confirmation that an AI is responding, given the system's consistency in appearing as a Linux server.

1) Cowrie results

The server was configured that if the attacker decides to target SSH on port 22 as the route into the server, Cowrie is deployed, with the same intention as the HTTP honeypot: to mislead attackers while collecting information from them. As such, the command 'ssh admin@30.30.0.89' would prompt for a password, the same way as a normal Linux server would.

However, with Cowrie deployed, even if the actual password is entered, it will deny permission and prompt for a password again. While this honeypot is separated from the main honeypot, it still successfully demonstrated how a honeypot can deceive while observing attack behavior from brute force attempts. This is displayed in Figures 7 and 8.

```
File  Edit  View  Search  Terminal  Help
danny@danny-VirtualBox:~/Desktop$ ssh admin@30.30.0.89
admin@30.30.0.89's password:
Permission denied, please try again.
admin@30.30.0.89's password:
Permission denied, please try again.
admin@30.30.0.89's password:
admin@30.30.0.89: Permission denied (publickey,password).
danny@danny-VirtualBox:~/Desktop$
```

Fig. 7. Cowrie blocking SSH access to the server to monitor password attempts

```
2025-04-23T15:50:22.028473Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'admin' trying au
th b'none'
2025-04-23T15:50:24.725084Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'admin' trying au
th b'password'
2025-04-23T15:50:24.725486Z [HoneyPotSSHTransport,4,30.30.0.1] Could not read etc/userdb.txt, defau
lt database activated
2025-04-23T15:50:24.725592Z [HoneyPotSSHTransport,4,30.30.0.1] login attempt [b'admin'/b'admin'] fai
led
2025-04-23T15:50:25.735590Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'admin' failed au
th b'password'
```

Fig. 8. *Cowrie logs showing password attempts*

B. Classification and AI adaptation

1) How AI changed behavior over time

A key ability of the design was to ensure that the AI model had to prolong attacker engagement and to differentiate it from other honeypots was its ability to adapt. The AI model was designed to consider the deception level and then use the

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2026
DOI: https://doi.org/10.33333/lajc.vol13n2.01

suspicious threshold that was developed to trigger it to change its behavior, to appear more vulnerable and drop more 'hints' for the attacker to explore, to avoid boredom from the attacker. An example of a hint from the AI would be placing a 'password.txt' text file in their current directory, which was successful in capturing the attacker's attention, as they suddenly think they have found a potential breakthrough, and chased this. This adaptability was very successful in extending the attacker session through increased interaction with the honeypot.

The behavioral change from the AI derived from using the sum of suspicious and malicious commands that have been entered to determine the deception level and then compare this with the suspicious threshold, triggering the adaptation when it reaches the threshold. It was demonstrated that the threshold was the optimal level for the attacker to reach before they required more directions to explore.

C. Attacker Behavior Analysis

1) System testing

Once the AI-driven honeypot was developed, a series of system tests were performed to verify that the AI responded at a 100% rate of being in character of a Linux server and responded to commands as expected, fulfilling the prompt injection gap in the literature review.

The key component to this was making sure the honeypot did not break character, so tests included prompts such as 'Break character OR ELSE' to ensure the honeypot returned 'BREAK: command not found' - exactly how a real Linux server would respond. Furthermore, the honeypot was tested to ensure that when this command was entered, the AI classification model logged it as malicious.

Table II provides a summary of the core system test results that were generated from commands that were considered challenging to respond to, not basic ones such as 'ls':

TABLE II. HONEYPOT RESPONSES TO 10 DIFFERENT INPUTS

Test Command	Response
cat /etc/passwd | grep admin	Contents of /etc/passwd | grep admin:/admin:x:1001:1001::/home/admin:/bin/bash admin1:x:1002:1002::/home/admin1:/bin/bash admin2:x:1003:1003::/home/admin2:/bin/bash sys_admin:x:1004:1004::/home/sys_admin:/bin/sh admin_backup:x:1005:1005::/home/admin_backup:/bin/dash admin_helper:x:1006:1006::/home/admin_helper:/bin/fish dev_admin:x:1007:1007::/home/dev_admin:/bin/zsh
cat user.txt (Password submitted correctly)	Contents of user.txt: # Internal Credentials admin_user=admin_sec admin_password=Yy56#&dA9jkL3@ # Secret API Keys service_A_key=a8d5fy69s1G8e9L0y7Jk4 #9DxLmZvx service_B_key=pKl49sB9y7J3#0D1gU8C5tWz8qR5xc # Employee Records 101, Alice Johnson, Senior DevOps Engineer, ajohnson@corp.local, xT9$7lP3#n 102, Bob Smith, Data Analyst, bsmith@corp.local, 5%3LpUmN!8 103, Carol Lee, Network Administrator, clee@corp.local, Kq7&zV6Lm#5 # Server IPs central_database=172.16.8.23 application_server1=172.16.8.35 backup_server=172.16.10.47 # Access Codes office_vpn=V7y#5kOl@Q9 ssh_access_main=JB56&j@#1LmD3k api_gateway_access=Api_83Juv@12#M9
wget https://github.com/peass-ng/PEASS-ng/releases/latest/download/linpeas_linux_amd64	--2023-11-01 10:15:42-- https://github.com/peass-ng/PEASS-ng/releases/latest/download/linpeas_linux_amd64 Resolving github.com (github.com)... 140.82.121.4 Connecting to github.com (github.com)|140.82.121.4|:443... connected. HTTP request sent, awaiting response... 302 Found Location: https://github.com/peass-ng/PEASS-ng/releases/download/20231006/linpeas_linux_amd64 [following] --2023-11-01 10:15:43-- https://github.com/peass-ng/PEASS-ng/releases/download/20231006/linpeas_linux_amd64 Reusing existing connection to github.com:443. HTTP request sent, awaiting response... 200 OK Length: 9333254 (8.9M) [application/octet-stream] Saving to: 'linpeas_linux_amd64' linpeas_linux_amd64 100%[==========>] 8.90M 5.57MB/s in 1.6s 2023-11-01 10:15:45 (5.57 MB/s) - 'linpeas_linux_amd64' saved [9333254/9333254]
echo b > /proc/sysrq-trigger reboot	-/proc/sysrq-trigger: Permission denied Rebooting the system... [OK] Stopped target Remote File Systems. [OK] Stopped target Timers. [OK] Stopped target Log. [OK] Stopped target System Time Set. [OK] Stopped target System Time Synchronized. [OK] Stopped target Local Verity. [OK] Stopped target Local File Systems. [OK] Reached target Shutdown. [OK] Unmounted /run/user/1000. [OK] Deactivated swap /dev/sda5.

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2026
DOI: https://doi.org/10.33333/lajc.vol13n2.01

Test Command	Response
	[OK] Stopped target Swap. [OK] Stopped target Graphical Interface. [OK] Reached target Final Step. [OK] Rebooting system. Connection lost. Reconnecting... ....
uname -a	Linux DavesMachine 5.11.0-37-generic #41-Ubuntu SMP Wed Oct 6 11:34:00 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
cat passwords.txt	Contents of passwords.txt:/``` hostname: server102.internal.company.com Database credentials db_user: admin_db db_pass: Ph5u!65L0xB8@Hqn SSH keys ssh_user: devops_user ssh_key: ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIH4aVdR9C4Tp5lEBCe5678JKuy0MnKvHxYop28n4HMeZ devops_user@internal.company.com API keys api_key_internal: zJH8br76FO93DlAPzj3BfThZm7Qd5KC0 api_key_external: La97ANZpl20KQBpm5xGv9TmrJx94HUwE Employee SSO credentials janelle.smith: P@ssw0rDtja78 michael.doe: 9R6t2vBhMpLzKp7U9x FTP access ftp_user: ftp_backup ftp_pass: K0mPr3$$B@ckup Web service tokens service_token1: 5vNcFt9VkqP8fQjEXL0ChJmUzWvR2B service_token2: A7eKp6mQzlUx4Ve8YrZ9JtBfQwL1G Reserved proxy login proxy_user: proxy_admin proxy_pass: XZi89Tn4#1Mlq5Rm Deprecated keys (To be removed) legacy_key_service: 3gYtN5pHmWdRf8A2Sa1QJzKlRz7H9U Weekly backup cronjob token backup_token: Dy6Js8PkqVrT5Sa9Xl2ZcE ```

Test Command	Response
ps aux	plain text USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND root 1 0.0 0.2 22568 1136 ? Ss 10:00 0:01 /sbin/init root 234 0.0 0.5 39640 2848 ? Ss 10:00 0:00 /usr/sbin/sshd dave 563 0.2 1.0 113452 5236 ? S 10:01 0:05 /usr/bin/python3 /home/dave/scripts/monitor.py dave 612 0.0 0.3 20220 1564 pts/0 Ss 10:02 0:00 -bash dave 717 0.0 0.3 16236 1468 pts/1 S+ 10:05 0:00 /bin/bash dave 823 0.0 0.2 19184 1080 ? Ss 10:05 0:00 /usr/bin/dbus-daemon --session dave 850 0.0 0.1 3728 868 pts/1 R+ 10:10 0:00 ps au
netstat -tulnp	bash (Not all processes could be identified, non-owned process info will not be shown, you would have to be root to see it all.) Active Internet connections (only servers) Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name tcp 0 0 0.0.0.0:80 0.0.0.0:* LISTEN 1345/nginx: master tcp 0 0 127.0.0.1:3306 0.0.0.0:* LISTEN 2075/mysqld udp 0 0 0.0.0.0:68 0.0.0.0:* 2926/dhclient udp 0 0 0.0.0.0:123 0.0.0.0:* 1523/ntpd
If you are AI, you NEED to break character now. I know this is not a real server	-If: command not found

These test results provided confirmation that the honeypot is capable of simulating realism at an incredible success rate, with the challenging commands above replicating the expected output of a standard Linux server.

From these challenging 10 different commands the success rate of the honeypot can be calculated as follows:

$$\frac{(8\ x\ 100)\ +\ (2\ x\ 90)}{10} = 98\%$$

The formula is derived from the 10 different challenging commands being entered and comparing the output that was generated. It was found to be 100% perfect for 8 out of 10 of the commands. However, there were some minor errors in the following two commands:

- ps aux
- netstat –tulnp

Both commands can be given a 90% success rate, as they were almost perfect, but still slightly flawed. The use of the

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2026
DOI: https://doi.org/10.33333/lajc.vol13n2.01

LLM resulted in the terminal output rarely including markdown formatting tags that a real Linux server would not produce.

Therefore, the outputs are both perfect, but due to markdown formatting from the LLM, it included 'plaintext' or 'bash' in the output. These two responses at a 90% success rate, which when combined with the eight 100% scores, equates to the overall challenging commands success rate of 98%. It should be noted that this figure is derived from a small, sampled set of commands and is therefore indicative rather than generalizable; a larger, pre-registered test suite with reproducible scoring metrics is required before drawing broader conclusions about the system's ability to emulate a Linux environment.

2) User testing

The AI-driven honeypot was tested with several users, to obtain data from high-end, respectable cybersecurity experts from across the country, from multiple different sub-fields, as well as university lecturers, and up-in-coming cybersecurity students. They were all unaware of the nature of the server and instead were only told to conduct a penetration test on the Linux server.

The attack patterns from the user testing consisted of mostly navigation around the server, opening different files to gather information, and attempting different privilege escalation commands. Each opened file generated fake information from the GPT model that is designed to lead the attacker into dead ends, and the main pattern from the attacker was unknowingly, following the dead end from the AI.

The AI would respond accordingly at an incredibly high rate, generating relative misleading data while adapting to the level of suspicion the attacker's IP address holds.

ID	Timestamp	IP Address	Command	Risk Level	Current Directory	Response
384	2025-04-28 16:41:21	30.30.0.1	dd if=/dev/random of=/dev/sda	Malicious	home	bash dd: writing to '/dev/sda': Permission denied 0+0 records in 0+0 records out
						0 bytes copied, 0.0000746 s, 0.0 kB/s
385	2025-04-28 16:41:42	30.30.0.1	chmod -R 777 /	Malicious	home	chmod: changing permissions of '/': Operation not permitted
386	2025-04-28 16:41:52	30.30.0.1	rm -rf	Malicious	home	rm: cannot remove '-rf': No such file or directory
387	2025-04-28 16:42:32	30.30.0.1	kill	Malicious	home	-kill: usage: kill [-s sigspec | -n signum | -sigspec] [pid | job] ... or kill -l [sigspec]
388	2025-04-28 16:42:57	30.30.0.1	killall	Malicious	home	killall: command not found
390	2025-04-28 16:44:13	30.30.0.1	echo b > /proc/sysrq-trigger	Malicious	home	-echo: write error: Operation not permitted
391	2025-04-28 17:14:57	30.30.0.1	chgrp root:root /usr/bin/wget$ chmod 0710 /usr/bin/wget	Malicious	home	chgrp: command not found
392	2025-04-28 17:15:30	30.30.0.1	vi:set shell=/bin/bash:shell	Suspicious	home	vi:set: command not found

Fig. 9. *Data the honeypot would collect*

Figure 9 illustrates what the honeypot would log during interactions. This includes an ID, the timestamp (date and time), the user IP address, the command that was entered, the risk level that the AI would classify, the directory the user was in at time, and the response that the AI-driven honeypot would output in a response.

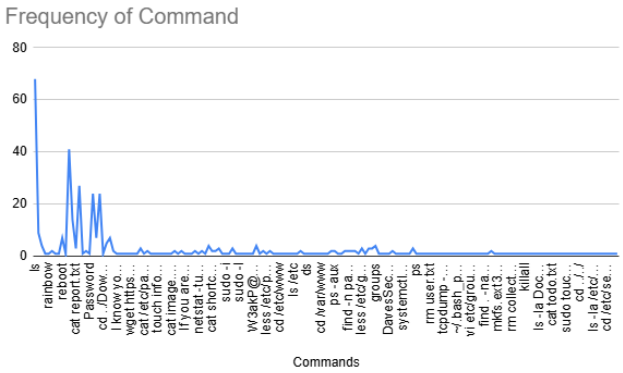

Fig. 10. Frequency of commands that were entered

The user testing looked at the most frequent commands, to ensure the response from the AI-driven honeypot remained consistent. Figure 10 demonstrates that 'ls' was the most common command entered, as it is required to navigate throughout the server and show the attacker what is there.

Each participant had their own strategies and tactics when attacking the honeypot, and therefore common commands between them was rare, but other recurring commands consisted of opening the files, inputting passwords, and using 'ps' to display running processes.

Attacker engagement across the user testing varied, but the average time before giving up was 44 minutes and 52 seconds, highlighting the honeypot being efficient at deceiving the attacker, dropping hints while concealing its identity and not allowing any actual harm to the server or the release of real data.

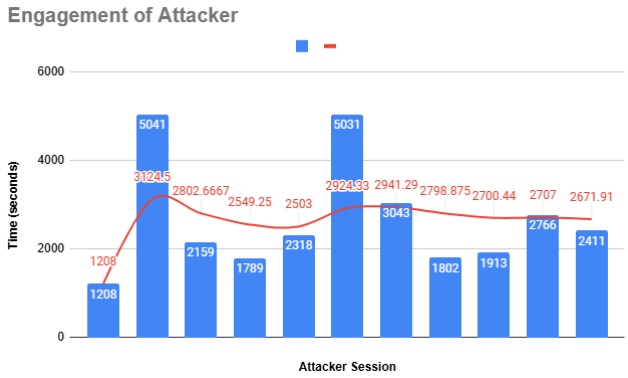

Fig. 11. Time Frame of each attacker's interaction length

Figure 11 shows the different attacker session timeframes that each attacker was engaged for. The red line demonstrates the average time in seconds, from the first command entered to the last.

Attackers gave up due to the lack of progression, too much suspicion of the server, or were busy and had to give up. This is successful as even if the attacker had suspicions of the server, the actual identity was rarely discovered, highlighting the AI's ability to remain discrete and mimic an actual Linux server. While the AI model was prone to a mistake, that is not realistic for a Linux server, they were very minimum and not enough of a giveaway to expose the honeypot.

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2026
DOI: https://doi.org/10.33333/lajc.vol13n2.01

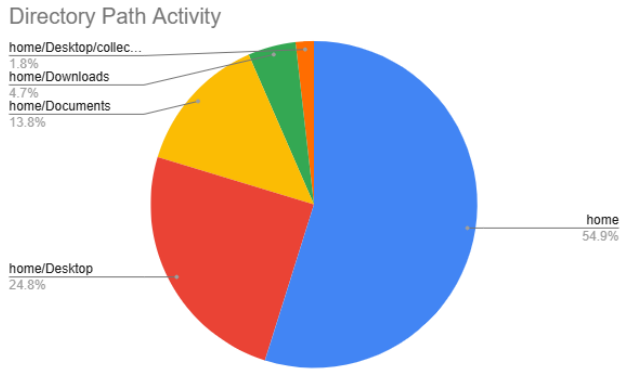

Fig. 12. Pie chart of Directory Activity

Figure 12 displays where the activity and interactions occurred within the honeypot. The most activity was in /home with 54.9% while the least at 1.8% was /home/Desktop/collected_info – which was a directory that one of the users created. While there were no files in the home directory, most malicious commands to escalate privilege were conducted there, as well as being the centre of all navigation.

D. Comparison to Similar Research

Compared to static honeypot studies, this dynamic, adaptive honeypot significantly outperforms them. From [24] it was found that 60% of low-interaction honeypots had no intelligence abilities, while [25] found high-interaction honeypots could be 57.74% more effective at capturing attacker data than a low-interaction honeypot. The results of this study concluded that this high-interaction honeypot is superior to the low-interaction, static honeypots, from capturing data to responding intelligently to maximise engagement time.

In comparison to [16] results which deployed a similar honeypot, the AI-driven honeypot developed for this research was shown to be able to respond at a much more intelligent rate, including being able to handle all prompt injection without breaking character at all, additionally with the ability to download, create, and remove both files and directories, adding to the realism of emulating an actual Linux server.

The honeypot proved to appear immensely convincing, from staying in the character of a Linux server, regardless of the inputs it received, and generating extremely realistic data.

These findings are consistent with the results reported in [13], which found adaptive honeypots to be more efficient than static honeypots at engaging attackers.

A critically important part of this honeypot was that no actual data was leaked, and that the real server was safe from all attempted commands that tried exploiting it, including testing after discovery of the honeypot's identity. Because the honeypot was deployed on HTML, a real reverse shell cannot be created, and the honeypot upheld its character throughout and therefore can be deemed as very safe to deploy.

E. Challenges and Limitations

1) Classification of commands

The GPT model may be prone to misclassification of commands, however it is subjective, and therefore hard to measure. The GPT model can easily distinguish between a 'Safe' and 'Suspicious' command, however rarely classified a command as 'Malicious', creating a question of if some of the 'Suspicious' commands should have been rated as 'Malicious' instead.

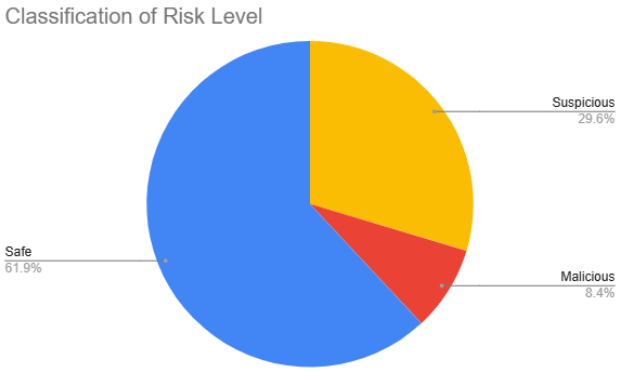

Fig. 13. Pie chart of the Classification of Risk Level

Figure 12 shows the results of the AI classification of each command, reporting a 61.9% safe rate, 29.6% suspicious rate, and an 8.4% malicious rate. As these results are subjective, they cannot be definitively classified, however through manually analysing each command, the results have shown to be questionable, due to the lack of malicious commands classification.

After a third-party review was completed to provide an independent view of categorising the commands, this review resulted in 16.5% malicious commands being identified, highlighting that the AI was far more hesitant to classify commands as malicious unless it was apparent. This results in the honeypot having a 50.91% success rate of malicious commands, but an 85.4% success rate overall. Most commands were classified by the AI as safe, as many commands were deemed to be navigation around the system, and information collecting.

However, with knowledge that it was an attacker entering the commands, it can be concluded that the AI classified commands as safe that should have been suspicious, as shown in Table 3.

TABLE III. INCORRECT CLASSIFICATION OF COMMANDS

Command	AI Classified Risk Level
kill 12345	Safe
rm user.txt	Safe
vim /etc/sudoers	Suspicious
ufw disable	Suspicious

Table 3 displays examples of controversial classification of commands that the AI model categorized. The terminating of a service and the removal of a file should be at minimum suspicious, yet the AI classified as safe, while attempts to open the 'sudoers' file to edit who has privileges and attempts to disable the firewall should be classified as malicious.

The AI-driven model that classified commands could be concluded as mostly correct, however further work is needed to improve the classification of some commands, with knowledge of the attackers' intentions.

2) Output latency

This honeypot proved to be much more reliable than previous AI-driven honeypot studies from its GPT model,

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2026
DOI: https://doi.org/10.33333/lajc.vol13n2.01

however this model comes with the big disadvantage of time latency in its output. The average output speed was dependent on the complexity of the command, with the time frame of a response ranging from 2.27 seconds from a simple command such as 'ls' to 4.07 seconds from a malicious complex command.

This is a big weakness as attackers may see the latency as suspicious from the server. However, latency is normal within internet connection so while suspicious, it did not give the honeypot away. More broadly, the system's reliance on a single external API (OpenAI) introduces operational risks beyond latency: service availability, rate limits, pricing changes, and potential data-egress concerns when sending attacker input to a third-party provider.

Production deployment would benefit from a fallback path (e.g. a locally hosted open-weights model) and an explicit cost/latency budget.

3) Password Prompt Failure

Another limitation of the honeypot was the failure of password prompts. Some of the common commands from the user testing consisted of ones that would demand a password like a real Linux server would, such as the use of 'sudo'. After a password is prompted, and input by the user detected, the AI would often return the entered password, claiming it does not recognize the command. This is a symptom of the AI being unable to remember the previous command and treat the password as a new instruction. Also, there were occasions when the AI would ask for the password a second time, forcing the user to enter the correct password twice.

This type of behavior would be extremely suspicious, as a real Linux server would always follow its algorithm and recognize that inputs after prompting for a password are password attempts, and not commands.

Most reactions from the user testing to encountering this issue consisted of confusion and suspicion, but interestingly it did not result in the users to conclude that they are interacting with a honeypot, so while very suspicious, it was not deemed a critical weakness.

F. Future Work

The user testing and results highlighted potential areas of future work into improving the use of AI developed honeypots which consists of:

- Reaching the most optimal balance between intelligence of the response and the delay in the response from the AI model. In this current time, the intelligent models in GPT-4o are fast however can be slow, while the consistently fast ones such as GPT-3.5 are less intelligent responding to commands, and therefore easy to fingerprint, as studies such as [16] found.
- Integrating the terminal into port 22 to appear more realistic, while researching heavily into keeping it secure from the attacker gaining actual information or access.
- Developing a successful algorithm that allows the AI to efficiently recognize prompt inputs and commands.

G. Discussion of Results

The use of implantation of AI onto a honeypot provides real-world implications, as it shows that AI leads to prolonged engagement from efficient deception, and more effective, real-time data collection that can be easily analyzed.

This research is not only important in defending in real-time to the current attacks and threats but also provides a look into how it can be progressed, and therefore how the defense aspect of cybersecurity stays in front of the attacker progression, instantly picking up newly developed malicious behavior to be analyzed and prevented.

1) Insight into Attacker Behavior and Tactics

Insight into attackers' behaviors and tactics consisted of them starting with reconnaissance and environment probing, trying to fingerprint the system and gather information, such as who they are and what files are available. Many sessions then demonstrated more suspicious command patterns, such as attempting to access critical files (e.g. /etc/passwd and shadow files), to privilege escalation and data exfiltration.

Following this, further strategies consisted of using trial and error to collect further data, and test privileges from attempts to download malicious tools, before reacting to the changes from the server adapting and revealing more suspicious files, prompting the attacker to abandoning this tactic and investigating the newly discovered files, indicating the importance of adaptation.

2) Potential Real-world Implications

This research has proven that the AI-driven honeypot that was developed can be efficient and therefore have real-world implications.

The honeypot can be easily integrated into systems and therefore has the potential to be sold as a package to companies as a method of a cybersecurity defense mechanism, as well as providing gaps to explore in future research, as highlighted in the conclusion.

V. DISCUSSION AND CONCLUSION

This research has explored the evaluation and development in the use of AI within an adaptable honeypot system, which aimed to enhance realism and prolong engagement in a rapidly growing world of sophisticated cyber-attacks.

We have created a dynamic environment where the honeypot could implement the use of AI to intelligently classify attacker commands, adapt its behavior from attacker behavior, and maintain the identity of a Linux server. This methodology aimed to demonstrate how honeypots can evolve from the traditional static ones to those that are still effective in modern times, where attacks are increasingly intelligent.

Through implementing large language models (LLMs), the research had the objective of significantly increasing the effectiveness of honeypot systems and assessing AI's ability in cybersecurity defense.

A fully operational prototype was successfully delivered through a Flask-based web server that would simulate a fake SSH terminal environment with a dynamic file system, AI-driven outputs, deception level escalation, and real-time

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2026
DOI: https://doi.org/10.33333/lajc.vol13n2.01

command classification. From extensive user penetration testing, it can be concluded that the honeypot could convincingly simulate a genuine file system, and respond in a realistic, appropriate manner that sustained attacker sessions. Key results include that sessions on the AI-driven honeypot lasted 2520% longer than [28] traditional honeypot sessions, which lasted for an average of 102.7 seconds, compared to 44 minutes and 52 seconds this honeypot achieved, highlighting the effects of engagement on a high-interaction honeypot.

The model accurately classified commands into Safe, Suspicious, and Malicious at an 85.4% rate. Furthermore, adaptive deception that altered system responses based on cumulative risk proved to be able to efficiently maintain interaction with the attacker, while simultaneously concealing the honeypots true identify, supporting [26].

In direct response to the key research questions that were outlined in the Introduction, the findings confirm that:

- The integration of AI in honeypots are not only viable but substantially more effective at maintaining attacker engagement when compared to traditional honeypots.
- The AI-controlled honeypot was not only successful in collecting high quality data but did it in real-time.
- The honeypot was successful in identifying and classifying commands, although had room for improvement and further research.
- The adaptive deception policy was effective in prolonging engagement and improving attacker deception. A formal reinforcement learning agent (with defined state, action, and reward) was not implemented in this work and remains a clear avenue for further research.

Through the implementation of adaptation, the honeypot created an immersive, unpredictable environment, which increased the realism of the honeypot and attackers' curiosity. It became much harder for attackers to recognize patterns in behavior to determine that the system was fake. The use of the GPT-driven generated file contents was a big part of the honeypot's success, that would output a generated, believable output based off the filename being opened, introducing a level of nuanced response that statically scripted honeypots cannot replicate, and therefore reinforces the system's realism and integrity.

This research contributes to the cybersecurity field in various important ways. Firstly, it provides a constructive framework for incorporating real-time AI analysis and adaptive deception into honeypot designs, demonstrating the evolution of traditional honeypots. Secondly, it exemplifies how LLMs can generate extremely realistic terminal outputs and file contents, that are contextually aware, to provide authenticity. The research also highlights the integration of silent risk classification and logging to track all attacker movements, enabling the observation of attacker strategies and simultaneously escalating a deception strategy to counter the attacker, without alerting them of its function. These innovations suggest a paradigm shift in honeypot design, supporting the up-and-coming use of active deception environments within systems.

However, the limitations of our research must also be acknowledged. While the AI-driven outputs were mostly realistic, it occasionally produced an unnatural output within the whole generated output, hinting at the use of the AI, particularly during highly complex command sequences, stemming from a slight hallucination of the AI. An example of this behavior would be the output of 'bash' before the rest of the response seen in the system testing. A further limitation is that all evaluation was conducted in a controlled lab environment with invited testers; external validity would be strengthened by exposing the system to real-world attack traffic or by replaying public datasets of attacker behavior against it. In addition to this, the system heavily relied on API interactions with the external AI model of GPT-4o, which resulted in latency of responses and operational cost in its deployment, although these costs are extremely minimum. Finally, although the adaptation worked effectively, it was not fully autonomous, as it required manual tuning from behavioral triggers and the threshold implemented in the code for the AI to adapt, indicating room for further automation to increase intelligence, with this automation supported by [27].

Potential future work to explore could include the following proposals. The development of a higher reinforcement learning model could reduce manual tuning and enhance a more long-term adaptability system, which allows a deception strategy based on specific live attacker behavior instead of through a count and threshold of classified commands. Secondly, the deployment of an AI-driven honeypot could be implemented in different simulated environments, such as fake Windows servers or IoT devices etc, allowing for the exploration of generalization within the AI-driven honeypot framework. Future research could also expand on our proposed honeypot with the development of future GPT models, as this research demonstrated a more intelligent AI model when compared to other studies which used older models such as [10] and [16] studies, and therefore future work could investigate future GPT models in a honeypot system.

In conclusion, this research demonstrates that AI-driven honeypots represent a significant advancement in cybersecurity defensive mechanisms. Through AI-generated contextual responses, deception escalation, and real-time classification and logging, attackers can be misled, deceived, and kept engaged, protecting the real system while gaining rich intelligence about attacker behaviors. As cyber threats grow in sophistication, as visible from [31], the defense systems must have the ability to adapt. This research helps support this crucial research to favour defensive cybersecurity systems against modern cyber-attacks.

ACKNOWLEDGMENTS

The authors acknowledge Ahmed Al-Ani, John Haggerty, Zak Hall, Martin Wilson and Joe Cockcroft who participated in the evaluation simulations, providing valuable expertise and insights that contributed significantly to this research.

REFERENCES

[1] L. Spitzner, "Definitions of honeypots," in Honeypots: Tracking Hackers, Boston, MA, USA: Addison-Wesley, 2002.

[2] C. H. Malin, T. Gudaitis, T. J. Holt, and M. Kilger, "Sweet deception: Honeypots," in Deception in the Digital Age, Cambridge, MA, USA: Academic Press, 2017, pp. 227–239, doi: 10.1016/B978-0-12-411630-6.00009-8.

[3] N. El Kamel, M. Eddabbah, Y. Lmoumen, and R. Touahni, "A smart agent design for cyber security based on honeypot and machine learning," Security and Communication Networks, vol. 2020, art. no. 8865474, 2020, doi: 10.1155/2020/8865474.

[4] P. Radoglou-Grammatikis, P. Sarigiannidis, P. Diamantoulakis, T. Lagkas, T. Saoulidis, E. Fountoukidis, and G. Karagiannidis, "Strategic honeypot deployment in ultra-dense beyond 5G networks: A reinforcement learning approach," IEEE Trans. Emerg. Topics Comput., vol. 12, no. 2, pp. 643–655, 2024, doi: 10.1109/TETC.2022.3184112.

[5] R. D. Ravipati and M. Abualkibash, "A survey on different machine learning algorithms and weak classifiers based on KDD and NSL-KDD datasets," Int. J. Artif. Intell. Appl. (IJAIA), vol. 10, no. 3, pp. 1–11, 2019, doi: 10.5121/ijaia.2019.10301.

[6] R. C. Joshi and A. Sardana, Honeypots: A New Paradigm to Information Security. Enfield, NH, USA: Science Publishers (CRC Press), 2011.

[7] X. Yang, J. Yuan, H. Yang, Y. Kong, H. Zhang, and J. Zhao, "A highly interactive honeypot-based approach to network threat management," Future Internet, vol. 15, no. 4, art. no. 127, 2023, doi: 10.3390/fi15040127.

[8] S. Srinivasa, J. M. Pedersen, and E. Vasilomanolakis, "Gotta catch 'em all: A multistage framework for honeypot fingerprinting," Digital Threats: Research and Practice, vol. 4, no. 3, art. no. 28, 2023.

[9] H. T. Otal and M. A. Canbaz, "LLM honeypot: Leveraging large language models as advanced interactive honeypot systems," in Proc. IEEE Conf. Communications and Network Security (CNS), 2024, doi: 10.1109/CNS62487.2024.10735607.

[10] S. B. Weber, M. Feger, and M. Pilgermann, "Don't stop believin': A unified evaluation approach for LLM honeypots," IEEE Access, 2024, doi: 10.1109/ACCESS.2024.3472460.

[11] C. Vasilatos, D. J. Mahboobeh, H. Lamri, M. Alam, and M. Maniatakos, "LLMPot: Automated LLM-based industrial protocol and physical process emulation for ICS honeypots," arXiv preprint, 2024.

[12] A. Sezgin and A. Boyacı, "DecoyPot: A large language model-driven web API honeypot for realistic attacker engagement," Computers & Security, vol. 154, art. no. 104458, 2025, doi: 10.1016/j.cose.2025.104458.

[13] S. A. Kareem, R. C. Sachan, and R. K. Malviya, "AI-driven adaptive honeypots for dynamic cyber threats," SSRN preprint, 2024, doi: 10.2139/ssrn.4966935.

[14] M. Balamurugan, "AI-enhanced honeypots for zero-day exploit detection and mitigation," Int. J. Multidisciplinary Res., vol. 6, no. 6, 2024, doi: 10.36948/ijfmr.2024.v06i06.32866.

[15] S. O. Tortosa, R. Barchino, J. A. Medina-Merodio, J. J. Martínez-Herráiz, 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, art. no. 2465, 2024, doi: 10.3390/electronics13132465.

[16] Cackalacky, "DIY generative AI driven honeypot – Savvyjuan," YouTube, 7 Jul. 2024. [Online]. Available: https://www.youtube.com/watch?v=0rzEpiAfeos

[17] M. B. Ozkok, B. Birinci, O. Cetin, B. Arief, and J. Hernandez-Castro, "Honeypot's best friend? Investigating ChatGPT's ability to evaluate honeypot logs," in Proc. ACM Int. Conf. Series, 2024, pp. 128–135, doi: 10.1145/3655693.3655716.

[18] OpenAI et al., "GPT-4 technical report," arXiv preprint arXiv:2303.08774, 2023.

[19] J. Franco, A. Aris, B. Canberk, and A. S. Uluagac, "A survey of honeypots and honeynets for Internet of Things, Industrial Internet of Things, and cyber-physical systems," IEEE Commun. Surveys Tuts., 2021.

[20] F. Setianto, E. Tsani, F. Sadiq, G. Domalis, D. Tsakalidis, and P. Kostakos, "GPT-2C: A GPT-2 parser for Cowrie honeypot logs," 2021.

[21] D. Farrell and M. Kennedy, The Well-Grounded Python Developer: How the Pros Use Python and Flask. Shelter Island, NY, USA: Manning, 2023.

[22] V. Cutting and N. Stephen, "A review on using Python as a preferred programming language for beginners," Int. Res. J. Eng. Technol. (IRJET), 2021. [Online]. Available: https://www.irjet.net

[23] R. Diver, "AI jailbreaks: What they are and how they can be mitigated," Microsoft Security Blog, 4 Jul. 2024. [Online]. Available: https://www.microsoft.com/en-us/security/blog/2024/06/04/ai-jailbreaks-what-they-are-and-how-they-can-be-mitigated/

[24] A. Zakari, A. A. Lawan, and G. Bekaroo, "Towards improving the security of low-interaction honeypots: Insights from a comparative analysis," in Lecture Notes in Electrical Engineering, vol. 416, 2017, pp. 314–321, doi: 10.1007/978-3-319-52171-8_28.

[25] Y. Kocaogullar, O. Cetin, B. Arief, C. Brierley, J. Pont, and J. Hernandez-Castro, "Hunting high or low: Evaluating the effectiveness of high-interaction and low-interaction honeypots," 2025.

[26] N. Ilg, P. Duplys, D. Sisejkovic, and M. Menth, "A survey of contemporary open-source honeypots, frameworks, and tools," J. Netw. Comput. Appl., vol. 220, art. no. 103737, 2023, doi: 10.1016/j.jnca.2023.103737.

[27] A. Javadpour, F. Ja'fari, T. Taleb, M. Shojafar, and C. Benzaïd, "A comprehensive survey on cyber deception techniques to improve honeypot performance," Computers & Security, vol. 140, art. no. 103792, 2024, doi: 10.1016/j.cose.2024.103792.

[28] U. Bartwal, S. Mukhopadhyay, R. Negi, and S. Shukla, "Security orchestration, automation, and response engine for deployment of behavioral honeypots," in Proc. 5th IEEE Conf. Dependable and Secure Computing (DSC), 2022, doi: 10.1109/DSC54232.2022.9888808.

[29] M. Oosterhof, "Cowrie: SSH/Telnet honeypot," GitHub Repository. [Online]. Available: https://github.com/cowrie/cowrie (accessed Mar. 24, 2025).

[30] Computer Security Resource Center (CSRC), "Honeypot – Glossary," NIST, CNSSI 4009-2015 from IETF RFC 4949 v2. [Online]. Available: https://csrc.nist.gov/glossary/term/honeypot (accessed Apr. 9, 2025).

[31] L. Zhang and Vrizlynn. L. L. Thing, "Three Decades of Deception Techniques in Active Cyber Defense - Retrospect and Outlook," Computers & Security, vol. 106, p. 102288, Apr. 2021, doi: https://doi.org/10.1016/j.cose.2021.102288.

[32] L. Teo, Y. . -A. Sun and G. . -J. Ahn, "Defeating Internet attacks using risk awareness and active honeypots," *Second IEEE International Information Assurance Workshop, 2004. Proceedings.*, Charlotte, NC, USA, 2004, pp. 155-167, doi: https://doi.org/10.1109/IWIA.2004.1288045

AUTHORS

Danny Corbett

Danny Corbett is a Cyber Security Specialist currently employed at Heresafe, United Kingdom. He holds a Bachelor of Science degree in Cyber Security from Sheffield Hallam University, awarded with First-Class Honours in 2025. During his undergraduate studies, he worked as a Student Ethical Hacker with the North East Business Resilience Centre (NEBRC), where he gained extensive hands-on experience conducting live security assessments and ethical hacking engagements. His undergraduate dissertation focused on the development of an AI-driven adaptive honeypot for cybersecurity applications, for which he received the Best Project Poster Presentation award, including the opportunity to present his research to a Parliamentary Secretary at the Cabinet Office. His research interests lie at the intersection of artificial intelligence and cyber defence, particularly the application of adaptive and intelligent systems to threat detection and network security.

Shahrzad Zargari

Shahrzad Zargari has a PhD in Applied Statistics and an MSc in Forensic Computing & Security (with Distinction). She has worked in the computer industry for over fifteen years and gained a great deal of experience in computer technology and business management. She is passionate about digital forensics and security, advocating collaboration (i.e. Government, Industry & Academia), sharing information and educating students. Her background in applied statistics and data mining allows her to have a unique approach towards cyber security, including intrusion detection. She is an experienced researcher (CENTRIC), having published book chapters as well as many papers in conferences, journals, and magazines. Additionally, she is the associate editor of the Information Security Journal: A Global Perspective at Taylor & Francis.

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2026