1International Journal of Innovative Research in Engineering
Volume 7, Issue 6 (November - December 2026), PP: 83-91.
https://www.doi.org/10.59256/ijire.20260706009
www.theijire.com
ISSN No: 2582-8746

Honeypot-Based Deception System for Automated Threat Detection

Srushti D. Sathe[1], Sharayu N. Patil[2], Diya C. Jangme[3], Swati A. Lade[4], Sakshi G. Nikam[5], Swati S. Patil[6]
[1,2,3,4,5,6] *D.Y. Patil College of Engineering & Technology Kolhapur, Maharashtra, India.*

How to cite this paper:
Srushti D. Sathe[1],Sharayu N. Patil[2], Diya C. Jangme[3], Swati A. Lade[4], Sakshi G. Nikam[5], Swati S. Patil[6], "Honeypot-Based Deception System for Automated Threat Detection", IJIRE-V7I6-83-91.

Copyright © 2026 by author(s) and Fifth Dimension Research Publication. This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/

Abstract: *Honeypots and cyber deception are well established methods of detecting suspicious activity and gathering information about attackers. Recent work has moved from traditional high-interaction honeypots toward systems built on LLM based deception systems that offer more realistic and context aware interactions [3], [4]. Autonomous AI agents have been identified as another form of cyber threat [5].Most existing approaches aims to make decoys look realistic, keeping attackers engaged, conserve resources or identify AI driven attacks. This paper proposes an AI Context Window Exhaustion Trap for AI-driven reconnaissance and asks whether carefully controlled deception can increase the contextual and computational workload of a reconnaissance agent while keeping the defender's server resources bounded. The proposed system combines request monitoring, synthetic data generation, suspicious session detection, an isolated deception endpoint, token and workload estimation and server resource tracking. When a session's suspicion score passes an estimated threshold then it is redirected to a controlled decoy environment containing fake or synthetic data. The study examines the response volume, reconnaissance success, estimated context load and defender resource use relate to one another. It aims to test whether context oriented deception can serve against AI-driven reconnaissance while keeping the system stable. This system is proposed as experimental design with no empirical results claimed.*

I.INTRODUCTION

The threat landscape is rapidly evolving and we need adaptive solutions. Web applications and networks are now facing a steady stream of scanners, bots and AI-driven agents which explore endpoints, collect available information, read responses and identify potentially useful resources, all without a human directing each step. Because this kind of automated reconnaissance mainly depend on detecting and blocking suspicious activities may not always be sufficient to handle such automated behaviour. Researchers explore defensive techniques which influence attacker activities and ensures the actual systems and crucial information remains protected.

Cyber deception is one of those approaches. Rather than only blocking intruders, it gives them fake systems, services, data or credentials that resembles genuine systems. Honeypots are designed to attract suspicious users by providing a simulated system that appears to be a genuine target. Any interaction with this system is monitored and stored in an isolated environment. The collected information helps security professionals identify unauthorized activities, examine attack techniques, and obtain useful information about potential threats [1], [2]. Honeypots are generally divided into three types: low interaction honeypots, medium interaction honeypots and high interaction honeypots according to the level of access and interaction they provide.At the high end, the attacker gets a fairly complete environment and the defender can capture keystrokes, network traffic and communication patterns. This level of detail is why high-interaction honeypots are favored for studying advanced attacks [2].A honeypot's efficiency is achieved only when its behavior is similar to that of a real system. If a decoy's responses or services are significantly different from those of a real application, attackers may notice such differences and realize that they are in a trap. They may therefore leave the environment, without continuing their activities. This limits the amount of useful information that can be collected. Alrubaish et al. stressed that realism must be assessed together with resource efficiency and proposed an evaluation framework based on behavioral fidelity, MITRE ATT&CK techniques, response timing, CPU usage and memory consumption [3]. Their work shows that the quality of deception cannot be judged independently of the computational cost of operating the defensive system.

AI and large language models (LLMs) have also changed how deception systems are built. Classic honeypots tend to be predictable, since their environments and responses rarely change. This has led recent work to turn to LLM-based honeypots, which generate responses on the fly and keep track of the conversation. They can imitate operating-system shells, network services and applications across many exchanges and are meant to be more realistic, hold attackers attention longer, gather more threat intelligence and adapt better than static decoys [4]. A parallel line of research treats autonomous AI agents as their own kind of adversary. One example is the LLM Agent Honeypot by Reworr and Volkov, which looks at interaction patterns, response latency and prompt-injection techniques to tell AI hacking agents apart from human attackers and ordinary bots [5].

Even so, most existing work centers on realism, adaptive interaction, engagement, AI-agent detection and low-cost deception. It is still a question whether purposeful misinformation can increase the contextual and computational burden on

an AI-driven reconnaissance agent. This does not attack the model itself, nor attempt to disable it. When suspicious reconnaissance is detected, the defender simply serves convincing but entirely synthetic content. By studying the resulting interaction, we can test whether the buildup of decoy material affects the agent's context-based workload and its real-world effectiveness.

This study explores the idea through a controlled and defense oriented model. The system combines a legitimate web application, request monitoring, suspicious-behavior detection, a controlled AI reconnaissance simulator, a deception endpoint, synthetic-data generation, token and context analysis, trap-session tracking and resource monitoring. The features for detecting suspicious behavior, include observable characteristics such as request frequency, burstiness, endpoint enumeration, sequential resource access, excessive pagination, repeated searches, authentication failures and session behavior . Once a configurable level of suspicion is reached, the session can be diverted to a deception environment populated with synthetic records rather than actual application data.

A key requirement is that the defensive server must remain resource-bounded. The system therefore does not rely on unrestricted generation or storage of decoy content. Streaming responses, bounded buffers, configurable record limits, session limits, rate limiting and cleanup mechanisms can be used to control the defender's resource consumption. This allows the experiment to measure the relationship between the decoy content delivered to the agent, the resulting token or context workload, and the resources consumed by the defender.

This study investigates whether raising the contextual and computational overhead of AI-driven reconnaissance agents through controlled deception can reduce their effectiveness without pushing the defender's server resource usage beyond set limits. This does not assume that all AI models are run out of capacity. We treat the context window running out as a hypothesis. This can be measured by looking at concrete metrics such as how many requests have occurred, responses received, tokens estimated, context extended, time taken, how far into the reconnaissance phase we are, CPU and memory usage, server stability. This model is intended for use on localhost, private laboratory environments or approved systems.

II.LITERATURE REVIEW AND RELATED WORK

2.1 Traditional Honeypots and Cyber Deception

Cyber deception is a proactive way of studying hostile activity inside a controlled environment. Honeypots are one of its most mature tools. They imitate real systems, services, or resources so that attackers are drawn in and their actions can be recorded. The data gathered this way feeds into threat intelligence, intrusion detection, network forensics and the study of attacker tactics, techniques and procedures (TTPs) [2]. Spitzner's early work showed how honeypots can be used to follow attackers and learn from their behavior [2]. Provos later introduced a virtual honeypot framework in which one host can emulate many network services and topologies, demonstrating that a large number of decoys can run without needing a dedicated physical machine for each [1].

2.2 High-Interaction Honeypots and Adaptive Deception

High interaction honeypots provide fully functioning systems and services. This means the attacker is not limited to only a few responses. Their aim is to keep attackers engaged for longer while collecting detailed information about their behavior, which makes them valuable for studying sophisticated threats [2]. The broader trend in the literature is a movement from fixed decoys toward adaptive deception that responds to what the interacting client does [6]. However, most of this work focuses on engaging adversaries and traditional threats and does not explicitly look at autonomous LLM-based agents and the ability to control their context driven workload with crafted deceptive content.

2.3 Realism, Fingerprinting and Resource Efficiency

A honeypot must be efficient and must not have features that allow an attacker to know it is a decoy. Alrubaish et al. investigated this problem in "Toward Realistic and Efficient Cyber Deception" and identified high-fidelity simulation as a central difficulty, because a decoy must reproduce the visible behavior of real services at reasonable resource cost [3]. They proposed a quantitative framework that evaluates realism using the MITRE ATT&CK framework, behavioral fidelity and resource-efficiency measurements, and compared honeypots across several network services with actual system implementations. Their evaluation considered realism, CPU utilization, memory usage, response time and fingerprinting weaknesses [3].

In addition, the efficiency of deception must be considered alongside the resource requirements of the honeypot. A honeypot that uses too much CPU/memory could be difficult to deploy or could affect performance in the environment being protected. In addition, honeypots that behave differently or at different speeds are easier to spot. This is directly relevant here, because the proposed trap must consider both the workload imposed on a reconnaissance agent and the resource consumption of the defender. The proposed system therefore treats defender resource use as an experimental indicator and does not assume that unlimited decoy content can be generated safely.

2.4 LLM-Based Cybersecurity Deception

Large Language Models have opened up new ways to make deception systems more adaptable and realistic. Singh's survey "Generative AI-Driven Cybersecurity Deception: A Brief Survey of Large Language Model Based Honeypots" examines how LLMs are being built into honeypots [4]. It names the static and predictable behavior of traditional honeypots as a major weakness because fixed responses make decoys easier to spot and hold attackers' interest less well. LLM-based systems address this by producing dynamic responses that take context into account. The survey discusses their use for simulating shells and network services as well as web applications and industrial control systems [4]. Prompt engineering,

in-context learning, chat history and fine-tuning. It notes that keeping session context lets the decoy answer consistently with earlier interactions [4]. Finally, it points to the potential of LLM-based honeypots for monitoring AI-driven attacks [4].Its focus, however, is on improving the deception environment itself. It does not propose deliberately increasing an AI attacker's contextual workload as a defensive mechanism, which is the point of departure for this study.

2.5 Honeypots for AI-Driven Reconnaissance

Autonomous AI agents form a new category of adversarial client. Unlike fixed scripts, an AI-driven reconnaissance agent can interpret responses, retain information from earlier interactions, choose subsequent actions, and adapt its strategy. Reworr and Volkov's "LLM Agent Honeypot: Monitoring AI Hacking Agents in the Wild" focuses on identifying and analyzing such agents, using interaction patterns, response latency, and prompt-injection-based techniques to separate them from human attackers and conventional bots [5].The work reflects a transition from conventional honeypot monitoring to deception systems aimed at AI-driven threats. Its focus remains on detection and observation, and it leaves open the question of using decoy information to raise an agent's context-based workload or limit its reconnaissance capability.

2.6 Cyber Deception Techniques to Improve Honeypot Performance

Javadpour et al. surveyed cyber deception techniques intended to improve honeypot performance, including approaches to resisting identification by attackers [6]. This survey presents a general foundation for understanding the advantages and drawbacks of defense using deception. However, it does not consider defense based on context window against LLM-based reconnaissance.

2.7 Comparative Analysis of Existing Research

Study	Main Focus	Major Contribution	Limitation Relevant to This Research
Provos [1]	Virtual honeypots	Simulate multiple hosts and services on a single system.	Pre-LLM agent based approach, does not focus on AI.
Spitzner [2]	Honeypot fundamentals	Explains concepts of honeypots including classifications, and their use in tracking attackers.	Does not specifically focus on AI.
Alrubaish et al. [3]	Realism and efficiency	Looks at realism, accuracy, fingerprinting and resource usage.	Does not specifically look at AI based reconnaissance agents.
Singh [4]	LLM-based honeypots	Discusses dynamic, context-aware deception across different environments.	Focuses on realistic interaction rather than deliberate increases in context workload.
Reworr and Volkov [5]	AI-agent honeypots	Investigates detection and analysis of potential autonomous AI hacking agents.	Focuses on identification and monitoring, not experimentally increasing agent workload.
Javadpour et al. [6]	Honeypot performance techniques.	This survey provides an overview of deception techniques that can be used to improve honeypot performance while making them harder to identify.	However, it does not focus on context window based techniques to defend against AI agents.
Proposed work	AI context-window exhaustion trap	Uses controlled synthetic-data delivery and measures observable workload and token-related metrics.	Requires experimental validation across controlled AI-agent configurations.

Table No:1

2.8 Identified Research Gap

The reviewed literature leaves several areas open. First, classic honeypot research provides strong foundations for deception and threat-intelligence collection, but its techniques were not designed for autonomous AI agents [1], [2]. Second, work on realism and efficiency offers useful evaluation principles for CPU, memory, response time, and behavioral fidelity, but it mainly assesses existing honeypots rather than proposing a mechanism aimed at shaping AI-driven reconnaissance [3]. Third, LLM-based honeypot research has greatly improved the ability to generate dynamic, context-aware interactions, yet it does not propose intentionally increasing an AI agent's context-window usage as a defensive mechanism [4]. Fourth, studies such as the LLM Agent Honeypot show that AI hacking agents can be observed through their interaction features, but their goal is identification rather than testing whether controlled decoy information increases an agent's contextual workload and affects reconnaissance effectiveness [5]. Finally, surveys of honeypot-performance techniques do not address this problem either [6]. This gap motivates the proposed AI Context-Window Exhaustion Trap.

III.PROLEM STATEMET AND RESEARCH QUESTIONS

3.1 Problem Statement

With increased use of AI and LLMs new issues in cybersecurity have arisen. AI based agents can be used for discovery tasks on web applications. They can be used to track resources that are available and to easily collect information. Traditional honeypots are used for anomaly detection, threat intel gathering and attack analysis. Recent advancements have increased realism by allowing dynamic engagement, AI generated responses and a cheaper way to deceive. The deliberate use of dynamically generated decoy information to increase the context-based workload of AI-driven reconnaissance agents, however, remains largely unexplored. This study therefore proposes an AI Context-Window Exhaustion Trap that detects suspicious automated activity and redirects it to a controlled deception environment containing dynamically generated synthetic information. The system aims to prolong the agent's interaction with decoy resources and to examine whether this reduces the effectiveness of automated reconnaissance while keeping the defender's resource consumption within predefined limits. The approach will be evaluated experimentally for its effectiveness in detecting suspicious behavior, engaging automated agents, and preserving the security and performance of the genuine application.

3.2 Research Questions

1. **RQ1**: How well can it detect automated reconnaissance attacks.
2. **RQ2**: Are dynamically created honeypot data able to increase the time spent interacting and increase the load on the AI based reconnaissance agents.
3. **RQ3**: How effective is the trap at mitigating the effects of an automated reconnaissance attack?
4. **RQ4**: How efficient is the system at managing honeypot interactions using little CPU usage, memory usage and response time?
5. **RQ5**: Can suspicious clients be redirected to the deception environment without impacting legitimate users or exposing the real application data.
6. **RQ6**: Compared to traditional honeypot solutions how effective is our solution at detecting reconnaissance attacks, using resources efficiently and engaging the automated reconnaissance agents.

IV.PROPOSED METHODOLOGY

This proposed system determines if controlled cyber deception can be used to affect the behavior of automated reconnaissance agents. The method uses behaviour monitoring, risk-based classification, artificial honeypots, and resource aware evaluation. Traditional honeypots are used to provide a controlled environment for monitoring suspicious traffic. Recent research has looked into using adaptive deception to increase engagement and realism [3], [4], [5]. This method is different to traditional detection methods which focus on identifying suspicious requests. The method has 5 components; collecting traffic data, analysing behaviour, classifying suspicious sessions, using controlled deception and evaluating performance.

4.1 Traffic Collection and Feature Extraction

Information is collected at this level, including the time, HTTP method, URL requested, status code, response time and size. Sessions are also logged where appropriate along with a small amount of meta information.

This provides insight into behaviour. Frequently hitting multiple admin resources, hitting many non-existent endpoints, or hitting a high number of requests in a short period could all indicate reconnaissance activity. However these features can be caused by legitimate testing and monitoring tools and so a combination of features are evaluated. This data is condensed down to provide session level features including, frequency, unique endpoints, percentage of failed requests, pattern of endpoints hit and overall data volume. This allows the session to be analysed as a whole. Reconnaissance is often made up of a number of related requests. Recording adversarial interactions in this way follows the established honeypot practice of logging attacker behavior [1], [2].

4.2 Behavioral Analysis and Suspicion Classification

Once the features have been extracted, we can evaluate them to provide a measure of how suspicious a session is. A rule based solution can be used to detect suspicious patterns like sequential endpoints being queried, reoccurring queries to seemingly sensitive paths and excessive request rates. These rules can be included at the beginning as they are easily explained and replicated.

A general suspicion score can be defined as:

$$S = w1.\ x1 + w2.\ x2 + \ldots + wn.xn$$

where S is the session suspicion score, x_i represents observations and w_i is the weight given to each observation. Weights should be learnt using labelled test traffic.

Once we have calculated the suspicion score we can classify sessions into normal, suspicious and high risk sessions. Normal sessions will flow through the normal application path, suspicious sessions are monitored further and high risk sessions could be diverted to a honeypot which is isolated, depending on our routing policy.

We could also add an anomaly detection system based on machine learning to give us more insight into how different features interact together. Anomaly detection and malicious-intent classification are not identical, however: an unusual session is not necessarily an attack, so model output should be interpreted alongside the available behavioral evidence. The effectiveness

of a honeypot also depends on its design, observability, and resistance to attempts at identifying the deception mechanism [3], [6].

4.3 Controlled Decoy Generation and Interaction

After suspicious behavior is identified, the system can give the session access to a controlled environment containing synthetic resources, such as fictional configuration files, simulated API documentation, dummy service inventories, and artificial links between related endpoints.

The decoy environment should be consistent enough to support continued interaction. For example, a simulated service description may refer to another fictional endpoint that returns a matching dummy response. This provides the environment for researching how the client interrogates the information.

The spoof information should never reveal any genuine credentials, production setup, personal information or sensitive business information. It should also be kept separate from the production application to ensure that the honeypot does not provide a vector to real systems. Earlier work on virtual honeypots demonstrated the role of consistent synthetic environments, including simulated network services and topologies, in cyber deception [1].

Research on LLM-based honeypots suggests that language models can support more adaptive, context-aware interactions than static decoys, although this flexibility brings additional evaluation and control requirements [4]. In this proposal we will take a slightly different approach. We will see what difference controlled interactions with decoys make by looking at how reconnaissance behaves.

4.4 Context-Window Workload Estimation

The central idea of the research is to investigate whether a decoy environment can increase the amount of information an automated agent must process before completing a reconnaissance task. This is motivated by growing interest in monitoring AI agents that perform security-related activities and by research on LLM-based cyber deception [4], [5].

The amount of information a server delivers, however, is not equivalent to the internal context usage of an AI model. In some cases agents may skip some information, condense previous information, shorten their responses or have a system which does not have all documents in context. The system should track metrics and should not assume longer responses will cause context window issues.

We will mainly track the number of requests made, how many decoys were visited, the amount of returned information, how many tokens this is likely to be and the time taken. If we have a system with an instrumented agent we can also track how many input tokens, output tokens and tool calls there were.

This can tell us if having decoys in the interaction increased the workload or changed the task performance. We should only make claims about context window issues if we have direct evidence. Without reliable agent-side telemetry, findings should be described as changes in observable interaction workload, not as proof that the model's context window was exhausted.

4.5 Resource-Aware Operation

Evaluate the cost of the deception system. It will log CPU usage, memory usage, latency, throughput and error rates. You should also impose some limits to prevent excessive amounts of data being returned, requests per session, session time and fake data. A deception system that causes excessive load on its own infrastructure is of little use. This stage follows the concern raised in research on honeypot realism and efficiency: Alrubaish et al. examine fidelity, fingerprinting, and operational efficiency [3], while Javadpour et al. survey techniques for improving honeypot performance and resistance to identification [6]. The proposed system accordingly evaluates both the potential benefit of deception and the computational cost of sustaining it.

V.SYSTEM ARCHITECTURE

The system would consist of 6 parts:

- Request Monitoring Layer
- Behavioral Analysis Engine
- Risk Classification
- Isolated Honey Pot
- Decoy Content Manager
- Evaluation and Logging Module

5.1 Request-Monitoring Layer

The monitoring layer is the entry point for requests reaching the experimental application. Collects information on incoming requests, and attempts to correlate the various events to one session. This is mainly to ensure consistency in the logging, whilst not affecting the performance of the application. It also makes sure that unnecessary sensitive information is not collected. Only the data required for the experiment is stored, and access to these logs is limited to authorized users. Session-oriented logging also supports analysis of interaction sequences over time [1], [2].

5.2 Behavioral-Analysis Engine

This engine will look at the patterns in the event logs to see if any could be indicative of a reconnaissance attack. Features to look out for would be high request frequency, high number of endpoints accessed, number of failed requests and

the sequence of resources accessed. Using a rule based detection system will be the first step in detecting, then we can also implement an anomaly detection model. This will help when trying to compare the different detection systems and understand what features affect the classifications. It is important to understand that suspicious behavior does not always mean that a malicious activity has taken place when analyzing automated detection results [6].

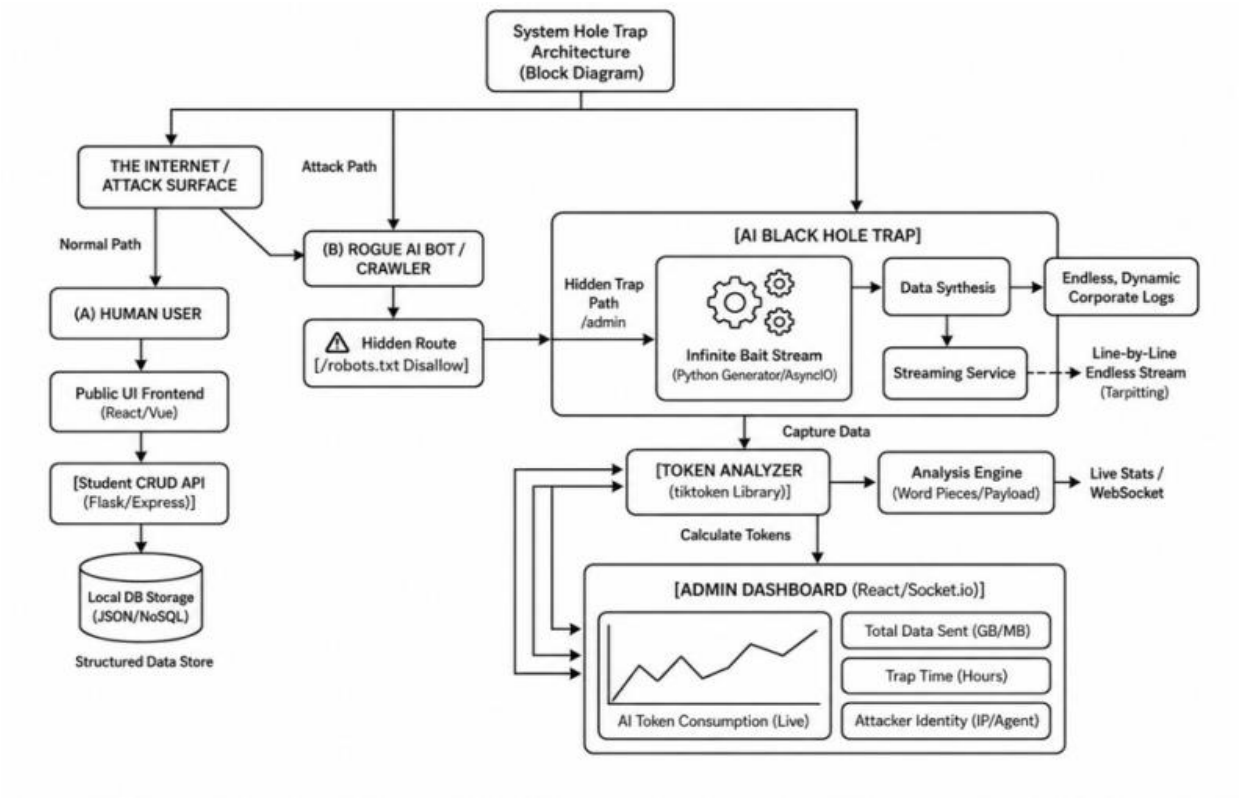


Fig5.1 System Architecture

5.3 Risk-Classification Module

This module turns the analysis results into an operational decision. This module will assign a risk category to the request, along with any supporting evidence. For example, if during a session there are a high number of unusual endpoints being probed quickly, alongside multiple attempts to request non-existent resources then this can be logged as a suspicious request. This decision should be based upon the pattern observed and not just due to one or two requests. This module should allow exceptions to be made for expected automated clients. This will ensure that testing/monitoring/development tools aren't affected by this decision. Evidence should be stored so that any wrong decisions can be analysed.

5.4 Isolated Honeypot and Decoy Content Manager

This module provides fake resources to certain sessions. A decision is made on what to provide based on what resource is requested using a pre-defined policy. The environment should keep relationships between simulated endpoints consistent, must not provide access to production services, and should bound response size, session duration, and the number of available decoy resources. Existing research describes how simulated services and virtual network environments can support deception, while more recent studies examine the relationship between realism, adaptability, and honeypot fingerprinting [1], [3], [4], [6]. The proposed implementation applies these principles to a limited web-based environment.

5.5 Logging and Evaluation Module

This provides session level data for analysis. It includes total number of requests, unique endpoints, volume of response data, time spent interacting with a system, classification outcome and resource usage. This allows comparisons between the standard and proposed configuration. The data can be used to analyze false positives, missed detections and differences between test clients. Comparable measurements are important because honeypot performance depends on both the behavior of the interacting client and the characteristics of the deception environment [3], [6].

VI.IMPLEMENTATION

The prototype can be developed using a light weight web framework such as Python FastAPI or Flask. The prototype will also use a relational database to store the session logs and summaries. Finally, a separate module will be used to perform the classification and calculation of metrics. The technologies used depend on the resources available and what was needed for the experimental setup. The prototype has been designed as shown below, however this should be updated to match the prototype being created.

6.1 request logging

Every request should be logged in a structured format. This should record details such as: time, session ID, HTTP method, endpoint, response status, response time, response size, risk score (optional), and if the request was dealt with by the normal application or honeypot. A consistent format is essential here as all the analysis will depend on the quality of the data. If the time is missing, session IDs aren't consistent or there are duplicates in the data, this can cause issues with the accuracy of the session lengths and number of requests [1] [2].

6.2 Detection & Routing

This module will calculate the behavioural features for a set observation period. The routing module will then decide if the session should continue to the normal application or be sent to the isolated decoy application based on the tuned classification policy. Every decision made should be logged with reasons why to allow for the classification to be validated against the testing labels. The evaluation should distinguish between detecting unusual activity and identifying confirmed malicious behavior [6].

6.3 Decoy-Content Management

The decoy content manager provides faked responses based on the requested resource. To begin with, it is best to use a template-based approach to give you more control and predictable resource usage. The content should be logically consistent but easily distinguishable from actual application content. Any extension using a language model to provide fake content should have constraints on the amount of content generated, safety mechanisms and secure them from being able to access internal services. Although LLM-based honeypots offer opportunities for adaptive interaction, their behavior and performance require systematic evaluation [4].

6.4 Workload and Resource Measurement

The measurement module records interaction duration, endpoint exploration, request count, response size, and estimated token volume where a compatible tokenizer is available. You could collect metrics on CPU, Memory, Latency and Throughput from your Server side. If you have an instrumented AI agent then you may also wish to collect metrics from it such as Token Usage and Tool Calls. However, you must ensure that these do not become confused with the estimates from the Server side. Research on AI-agent honeypots provides context for studying automated interactions, but it does not establish that the proposed system will exhaust an agent's context window [5].

6.5 Security Validation

The prototype should be tested before beginning the experiment to ensure the decoys are isolated, rate limiting is working and malformed requests cannot bypass the system, whilst also ensuring normal users aren't unnecessarily redirected. A failure state should be defined so if the system exceeds a certain number of resources being used, it will stop. All of these should be documented along with the testing environment and config so results can be accurately interpreted.

VII.EXPERIMENTAL DESIGN

The aim of this experiment is to test how the new context workload based deception compares to a standard honeypot. This is done to understand if the new approach will alter the behaviour of reconnaissance activities in a way that doesn't result in too much resource usage by the defenders. Due to how honeypots can be affected by the design of the honeypot itself, client activity and resource usage [3][6] comparison needs to be carried out in a controlled manner.

7.1 Experimental Environment

The prototype should be deployed within an isolated environment (local or virtual). This should include a monitored web application, fake decoy environment, traffic generator and resource monitor. The environment should be documented fully with information on what OS, processor, memory, application settings, database and software versions are being used to allow replication. Only authorized test clients and fake data should be used. If using any AI agents then they should be documented fully with what model they're using, tools available, instructions given and any relevant metrics. Different AI agents may behave differently depending on the content of the decoys. Previous work has been done on AI agent honeypots with a monitored environment to understand any potential AI activity [5].

7.2 Scenarios

Multiple scenarios should be created to ensure the system isn't tested in just one way.
Scenario 1 – Normal Users: This scenario uses simulated users browsing standard pages, this will give a baseline for the system and will show how many false positives there are.
Scenario 2 – Standard Automated: This scenario uses a test script to access a few authorized endpoints. This will show if automated activity is falsely identified as reconnaissance.
Scenario 3 – Reconnaissance: This scenario uses a controlled client to access numerous endpoints, including fake and non-existent admin routes. This will help to understand if the analysis engine can detect endpoint enumeration.
Scenario 4 – Decoy Testing: This scenario uses a controlled client to navigate between the fake resources. Metrics will be collected on how many requests were made, how many unique endpoints, how long and what was returned [1][6].
Scenario 5 – Instrumented AI Agent: If an authorized AI agent is available then it will complete a pre-defined task within the simulated environment. Metrics will be gathered on the actions taken, if the task was completed and any token/tool usage [4][5].

Scenario 6 – Resource Usage Testing: This scenario will test how the system deals with limited increases in requests. This will identify if there are any issues with high CPU/Memory usage, latency or errors due to the monitoring and deception. Each scenario should have predefined stopping conditions and be repeated under comparable conditions. The number of trials and the traffic-generation procedure should be reported in the final paper.

7.3 Baseline Comparison

Two primary configurations are proposed. The basic setup consists of a normal honeypot which only logs interactions and uses a fixed set of honeypot responses. The improved setup includes behaviour based classification, filtering, limited honeypot resources and workload measurements. An alternative setup containing only the detection component is also possible to determine if any improvements are due to improved classifications or the resulting deception.

All setups should have similar hardware, network conditions, traffic volumes and observation times. Differences in the size/contents of the responses should be noted as this can impact on client behaviour as well as defender resources. This follows the need to evaluate deception effectiveness together with realism and operational cost [3], [6].

7.4 Evaluation Metrics

The evaluation should cover both detection performance and operational efficiency. The detection rate is the proportion of suspicious test sessions correctly identified:

$$\text{Detection Rate} = TP / (TP + FN) \times 100$$

This represents the false positive rate of the system. It tells us how many real sessions were incorrectly identified as suspicious:

$$FPR = FP / (FP + TN) \times 100$$

Where TP = true positives, TN = true negatives, FP = false positives and FN = false negatives. Additional metrics include precision, classification latency, session duration, unique endpoints explored, follow-up request count, response-data volume, CPU utilization, memory consumption, and response latency. These metrics are proposed for the study and must not be presented as measured results until data have been collected.

Where a tokenizer or instrumented AI agent is available, token-related measurements may be added. The estimated token count of server responses must not be reported as the agent's actual context usage unless that usage is independently measured.

7.5 Data Analysis

Each scenario should be run multiple times. Descriptive statistics should be calculated (mean values for general trends, standard deviation and percentiles to show variability). Analysis should be done to determine whether results should be treated as individual requests or entire sessions. As requests will be dependent within a session this could lead to exaggerating evidence if treated as independent. If there are sufficient independent observations, appropriate statistical tests can be performed to compare different configurations. The type of test should depend on the distribution of the data and the experiment design. In the final paper it should be stated what tests were performed, what assumptions were made and what significance level was used.

VIII. RESULTS AND DISCUSSION

The results of this research must come from measurements taken on the implemented prototype. Because experiments have not yet been run, this section sets out the analytical structure for reporting results and does not contain numerical findings.

8.1 Detection Performance

The first part of the evaluation compares the classification performance of the standard and proposed configurations. Each scenario should be run multiple times. Descriptive statistics should be calculated (mean values for general trends, standard deviation and percentiles to show variability). Analysis should be done to determine whether results should be treated as individual requests or entire sessions. As requests will be dependent within a session this could lead to exaggerating evidence if treated as independent. If there are sufficient independent observations, appropriate statistical tests can be performed to compare different configurations. The type of test should depend on the distribution of the data and the experiment design. In the final paper it should be stated what tests were performed, what assumptions were made and what significance level was used.

8.2 Effect of Decoy Information on Interaction Behavior

The second part examines how test clients behave after meeting synthetic resources. The length of the sessions, number of requests made, unique endpoints accessed, and the volume of response data can be compared between both the baseline and new scenarios. An increase could show that the decoy environment is encouraging exploration; however this doesn't necessarily mean that the deception was successful. The client could have performed this exploration without being deceived or recognized the contents were fake and stopped. Having a specific aim to measure how effective the reconnaissance is would be useful. A controlled agent could be given a task to search for certain known facts within a simulated environment, and the accuracy, speed and information retrieval could be measured before and after.

References

1. N. Provos, "A Virtual Honeypot Framework," in *Proc. 13th USENIX Security Symposium (USENIX Security 04)*, 2004, pp. 1–14. [Online]. Available: https://www.usenix.org/conference/13th-usenix-security-symposium/virtual-honeypot-framework
2. L. Spitzner, *Honeypots: Tracking Hackers*. Boston, MA, USA: Addison-Wesley Professional, 2002.
3. H. Alrubaish, W. Aljoby, A. Aldeek, M. Alkubaish, and D. Mashima, "Toward Realistic and Efficient Cyber Deception," *IEEE Open Journal of the Computer Society*, vol. 7, pp. 574–586, 2026, doi: 10.1109/OJCS.2026.3669021.
4. V. K. Singh, "Generative AI-Driven Cybersecurity Deception: A Brief Survey of Large Language Model-Based Honeypots," in *2025 IEEE 2nd Int. Conf. Electrical, Electronics, Communication and Computers (ELEXCOM)*, 2025, doi: 10.1109/ELEXCOM67950.2025.11451325.
5. Reworr and D. Volkov, "LLM Agent Honeypot: Monitoring AI Hacking Agents in the Wild," arXiv preprint arXiv:2410.13919, 2024, rev. Feb. 2025. [Online]. Available: https://arxiv.org/abs/2410.13919
6. 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.
7. Ulka Madhukar Bansode, Sharayu Nilesh Patil et.al.,"Securing the Future of Health: A Survey on Data Protection and Privacy Techniques in a Decentralized System", Taylor & Francis,2025