

AI-Powered Cyber Deception System Using Dynamic Honeytokens for Credential Misuse Detection

Anitha V¹, Makheshkumar R P², Thirunithi T³, Saranraj G⁴, Anand S⁵

¹*Assistant Professor, Department of Artificial Intelligence and Data Science*

^{2,3,4,5}*Undergraduate Student, Department of Artificial Intelligence and Data Science*

¹anithakumaran29@gmail.com, ²makheshkumar2005@gmail.com, ³nithi6797@gmail.com,
⁴Saranrajvanitha2005@gmail.com, ⁵anand06.tech@gmail.com

Abstract—Modern security tools such as firewalls and intrusion detection systems struggle to detect attackers using stolen but legitimate credentials because their actions closely resemble normal user behaviour. This paper proposes an AI-powered cyber deception system that generates and manages dynamic honeytokens, fake credentials, and digital assets with no legitimate use to expose unauthorised access. The proposed architecture integrates AI-based honeytokens generation, a token-matching detection engine, real-time alerting, and a monitoring dashboard into one pipeline. As a conceptual, design-stage framework, it aims to enable early, near-zero-false-positive detection of credential misuse and lateral movement.

Keywords—Cyber Deception; Honeytokens; Credential Misuse Detection; Artificial Intelligence; Intrusion Detection

I. INTRODUCTION

Modern organisations increasingly rely on firewalls, intrusion detection systems (IDS), and endpoint security tools to defend their networks against cyberattacks. However, these tools largely detect anomalies in network traffic or known attack signatures, and they struggle when an attacker uses stolen but legitimate credentials such as usernames, passwords, API keys, or authentication tokens. Because the attacker's actions closely resemble those of an authorised user, such intrusions frequently evade detection until significant damage, including data exfiltration or lateral movement within the network, has already occurred.

Cyber deception has emerged as a complementary defence strategy that addresses this gap. Rather than attempting to distinguish malicious behaviour from legitimate behaviour after the fact, deception techniques place fake digital assets, or honeytokens, throughout a network that have no legitimate purpose. Any interaction with such an asset is, by definition, unauthorised, which allows deception-based systems to generate alerts with a near-zero false-positive rate. Deception technology has evolved considerably, moving from static, manually planted honeypots toward increasingly adaptive and AI-driven systems capable of generating context-aware decoys and analysing attacker behaviour in real time, a trend recently systematised for LLM-driven honeypots [8].

Recent research has substantially improved the realism and detection resistance of individually generated honeytokens, for example through large language model-based generation and organisation-context-aware credential synthesis. However, much of this work concentrates on the quality of token generation in isolation, with comparatively less attention paid to how such tokens should be generated, deployed, monitored, and escalated into actionable alerts as part of a single, deployable pipeline suited to practical security operations.

This paper proposes an AI-Powered Cyber Deception System that integrates AI-based dynamic honeytokens generation with a token-matching detection engine, real-time alert generation, event logging, and a security dashboard into one unified architecture. The system is presented as a conceptual, design-stage framework: it defines the architecture, workflow, and expected operational behavior of the proposed system, with empirical implementation and evaluation identified as future work. The remainder of this paper is organized as follows. Section II reviews related work in honeytokens generation and cyber deception. Section III presents the proposed system architecture and methodology. Section IV discusses the expected behaviour and a qualitative comparison of the proposed system against existing approaches. Section V concludes the paper.

II. LITERATURE SURVEY

The concept of honeytokens originated with Spitzner [1], who characterised deception assets into four broad categories, including fake user accounts, artificial files, dummy database records, and synthetic credentials. Juels and Rivest [3] later formalised honeywords, decoy password hashes stored alongside real password hashes, and demonstrated that an attacker who steals and cracks a password database cannot identify which entry is genuine without triggering a server-side alert. These early works established the foundational principle that underlies all honeytokens-based detection: any use of a decoy credential is inherently and unambiguously malicious.

Beyond specific mechanisms such as honeywords, Almeshekeh and Spafford [2] provide a broader theoretical treatment of cyber deception, characterising it as the deliberate manipulation of an adversary's perception and decision-making

process. This perspective underlies the design rationale for honeypot-based detection more generally, including the system proposed in this paper.

Recent work has focused on automating honeypot generation and improving its realism. Prabhaker et al. [5] proposed a framework for generating and deploying honeypots in relational databases, using a hierarchical machine learning algorithm that models parent-child relationships across multiple database tables to produce more convincing decoy records. Separately, Reti et al. [6] investigated the use of large language models to generate honeypots directly, showing that generative models can produce credentials that are difficult to distinguish from genuine ones using simple detection heuristics. Honeypot-based detection has also been applied beyond enterprise credential security: Farooqi et al. [4] embedded honeypots in social media account data to detect misuse of personal information by third-party applications, illustrating the generality of the honeypot concept across domains.

Sladić et al. [7] introduced VeLLMes, a high-interaction AI-based deception framework capable of simulating realistic system responses to attacker input, extending deception beyond static credentials into full interactive environments. Most recently, Weinberg [10] proposed PHANTOM, a framework that encodes organization-specific context, such as domain names, service naming conventions, and technology-stack idioms, into the honeypot generation pipeline, and introduced a formal believability scoring metric to quantify how convincingly generated tokens resemble genuine credentials. That work reports substantial gains in both human-evaluator acceptance and resistance to automated secret-scanning tools compared to static, template-based honeypot generators.

Broader systematisation efforts have also emerged. Bridges et al. [8] survey the intersection of honeypots and large language models, identifying a canonical architecture for LLM-driven deception and highlighting open challenges toward autonomous, self-adapting honeypot systems. At the architectural level, Yu et al. [9] proposed HoneyFactory, a container-based honeypot architecture that similarly aims to unify multiple deception components into a single deployable system, though its focus is on network-level honeypot orchestration rather than the credential-level honeypot generation, matching, and alerting pipeline addressed in this paper.

Collectively, this body of work demonstrates that AI-based generation techniques can meaningfully improve the quality and detection resistance of individual honeypots. However, these approaches are largely evaluated as standalone generation techniques rather than as components of a complete detection-and-response pipeline. Comparatively little published work addresses how dynamically generated honeypots should be deployed across an organisation, matched against live access attempts, and converted into structured, real-time security alerts through an integrated monitoring and dashboard layer. This is the gap that the present work seeks to address.

III. METHODOLOGY / APPROACH

The proposed AI-Powered Cyber Deception System is designed as an end-to-end pipeline spanning honeypot generation, deployment, detection, and response.

A. Architecture Overview and Block Diagram

Fig. 1 illustrates the proposed three-tier architecture. Tier 1 handles AI-based honeypot generation and deployment to controlled locations; Tier 2 performs continuous monitoring and token-matching-based detection; and Tier 3 handles alerting, event logging, secure storage, and dashboard visualisation for security personnel.

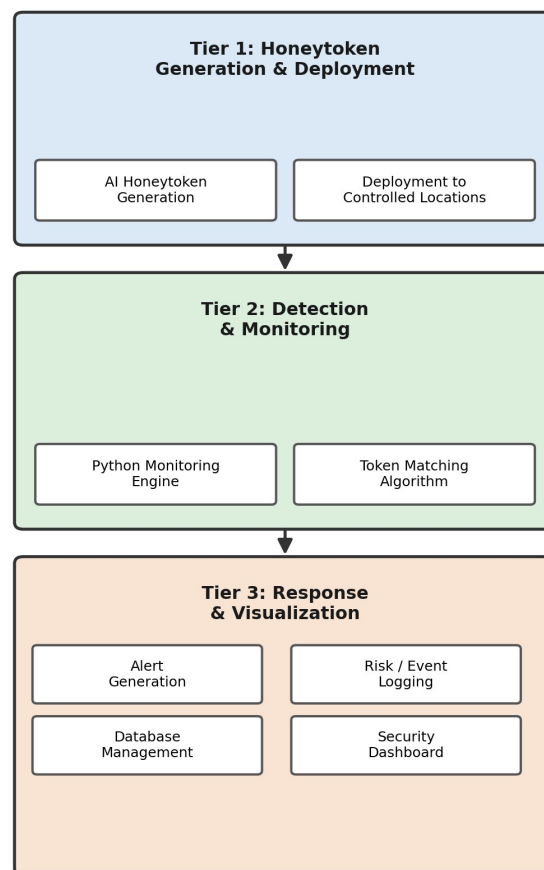


Fig. 1. Architectural block diagram of the proposed AI-Powered Cyber Deception System, showing the three-tier flow from honeypot generation and deployment through detection to response and visualisation.

The core modules of the proposed architecture, and their purpose are summarised below.

- **AI Honeypot Generation** — generates realistic, context-aware fake credentials (e.g., API keys, database credentials, cloud credentials, authentication tokens).
- **Token Matching Algorithm** — detects unauthorised use of deployed honeypots.
- **Python Monitoring Engine** — continuously monitors access attempts across deployed tokens.
- **Alert Generation** — produces real-time security notifications upon honeypot use.
- **Risk / Event Logging** — records suspicious activity details (IP address, timestamp, token ID, request information).
- **Database Management** — provides secure storage for tokens and incident records.

- **Security Dashboard** — visualizes active and historical incidents for security personnel.

In the proposed workflow, an administrator first uses the AI Honeytoken Generation module to create realistic fake credentials tailored to resemble the organisation's actual naming conventions and infrastructure. These honeytokens are then deployed by the system into controlled locations across the environment, for example, configuration files, source code repositories, or database tables, where a legitimate user would have no reason to access them.

A Python-based Monitoring Engine continuously observes access attempts across these deployed locations. When a request is made using a value that matches a stored honeytoken, the Token Matching Algorithm flags the event as unauthorized. Because honeytokens are never used in legitimate operations, this match is treated as a definitive indicator of compromise rather than a probabilistic anomaly score, which is expected to substantially reduce false-positive alerts compared to conventional anomaly-based intrusion detection.

Upon detection, the Alert Generation module records the associated metadata, including the source IP address, timestamp, matched token identifier, and request details, through the Risk/Event Logging component, and stores this information securely via the Database Management layer. A Security Dashboard then visualises active and historical incidents in real time, enabling security personnel to observe credential misuse and lateral movement attempts as they occur, and to correlate multiple honeytoken triggers to identify broader attack patterns.

B. Detection-to-Alert Pipeline Flowchart

Fig. 2 summarises the decision logic followed each time an access attempt is observed, from initial monitoring through token matching to alert generation and dashboard reporting.

As shown in Fig. 2, the pipeline begins when a honeytoken is deployed and registered in the secure token store. The Monitoring Engine then observes each credential-bearing access attempt across the protected environment and compares the presented value against the stored honeytoken registry. If no match is found, the request is treated as ordinary activity, and the engine simply resumes monitoring, adding no friction for legitimate users. Because honeytokens are never issued to real users or services, only a genuine match advances the request to the next stage.

On a match, the event is flagged as unauthorized and its metadata, including the source IP address, timestamp, token identifier, and request details, is logged to the incident database. A real-time alert is then generated and the Security Dashboard is updated so that analysts can review the incident and decide on a response, such as revoking related credentials or isolating the affected host. Repeated triggers from the same source can be correlated on the dashboard to reveal lateral movement, while the matching step itself is designed to remain a simple lookup, which keeps decisions easy to audit.

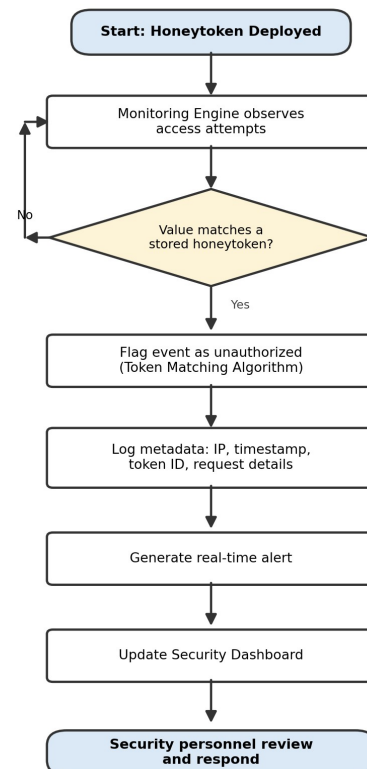


Fig. 2. Operational decision flowchart of the proposed system, showing the path from monitored access attempt to token match, alerting, and dashboard update.

As an illustration, consider a scenario in which an attacker gains access to a developer's misconfigured Git repository and extracts what appears to be a valid cloud API key. Because this key is in fact a dynamically generated honeytoken, any attempt to use it against the organization's cloud provider is immediately flagged by the Token Matching Algorithm, triggering an alert that identifies the token, the time of use, and the origin of the request, well before the attacker can pivot to genuine production systems.

IV. RESULTS & DISCUSSION

As the proposed system is presently at the design and architecture stage, this section does not report empirical performance measurements from a deployed system. Instead, it discusses the expected qualitative behaviour of the proposed architecture and situates it relative to existing honeytoken and deception approaches identified in Section II. Empirical validation through an implemented prototype is identified as future work in Section V.

A. Proposed Evaluation Environment: Network Topology and Segmentation

To guide the planned empirical validation described in Section V, this section outlines a proposed network segmentation strategy for the future evaluation testbed. Table I presents an example division of a mid-sized enterprise network into security zones, together with an illustrative honeytoken placement strategy for each zone. These are proposed design

choices for the planned testbed, not measurements from a deployed system.

TABLE I. PROPOSED NETWORK SEGMENTATION FOR THE PLANNED EVALUATION TESTBED

Zone	Purpose / Example Assets	Honeytoken Placement Strategy	Relative Priority
DMZ/Perimeter	Public-facing services (VPN, web/app gateways)	Decoy VPN credentials and API keys	Medium
Internal Workstations	End-user devices and shared drives	Decoy stored browser/credential-manager entries	Medium
Application/Database Servers	Internal business apps, databases	Decoy database connection strings and service credentials	High
Identity&Directory Services	Active Directory/LDAP, authentication servers	Honeywords and decoy privileged-account tokens	Critical
Data/OT Storage	Backups, sensitive data repositories	Decoy cloud storage keys and backup credentials	Critical

This segmentation follows common enterprise security-zone conventions and concentrates higher-priority honeytoken placement in zones with the greatest potential impact if compromised, such as identity infrastructure and sensitive data stores. Concrete node counts, traffic volumes, and criticality weightings for this topology are intentionally left unspecified here, since they will be defined during prototype implementation and no testbed has yet been deployed.

- **Static, manually planted honeytokens [1]:** Token realism/adaptability: low, fixed content is fingerprintable over time. False-positive rate: very low. Pipeline integration: manual monitoring; no automated matching or dashboard.
- **AI-generated tokens, generation-focused [6], [10]:** Token realism/adaptability: high, context-aware and detection-resistant. False-positive rate: very low. Pipeline integration: not addressed; focused on generation quality alone.
- **Proposed AI-Powered Cyber Deception System:** Token realism/adaptability: high, AI-generated and context-aware. False-positive rate: very low (expected). Pipeline integration: full — generation, matching, alerting, logging, and dashboard combined.

B. Comparative Analysis

Table I situates the proposed system qualitatively against static honeytokens and generation-focused AI approaches identified in Section II, based on the properties each approach targets by design rather than on measured benchmarks.

QUALITATIVE COMPARISON OF DECEPTION APPROACHES

Approach	Token Realism / Adaptability	False-Positive Rate	Pipeline Integration
Static, manually planted honeytokens [1]	Low — fixed content, fingerprintable over time	Very low	Manual monitoring; no automated matching or dashboard
AI-generated tokens,	High — context-aware,	Very low	Not addressed; focused on generation

generation-focused [6], [10]	detection-resistant		quality alone
Proposed AI-Powered Cyber Deception System	High — AI-generated and context-aware	Very low (expected)	Full — generation, matching, alerting, logging, dashboard

By combining AI-driven token realism with a dedicated matching, logging, and alerting pipeline, the proposed system is expected to retain the near-zero false-positive property inherent to honeytoken-based detection while additionally providing the operational tooling, real-time alerts, incident logging, and a centralised dashboard needed to act on detections quickly. This is expected to reduce attacker dwell time and limit lateral movement following credential theft, compared to approaches that address token generation alone.

The principal limitation of the present work is that it has not yet been implemented or evaluated empirically. Future work will involve building a working prototype, deploying it within a controlled testbed environment (e.g., using an existing honeypot platform such as Cowrie or T-Pot), and measuring concrete metrics such as detection time, false-positive rate on legitimate traffic, and attacker dwell time, to validate the qualitative expectations discussed in this section.

V. CONCLUSION

This paper proposed an AI-Powered Cyber Deception System that unifies AI-based dynamic honeytoken generation with token matching, real-time alerting, event logging, and dashboard visualisation into a single architecture aimed at detecting credential misuse and unauthorized lateral movement. Unlike much of the existing literature, which primarily addresses the realism of individually generated honeytokens, the proposed system emphasises an integrated, end-to-end pipeline intended for practical deployment within an organisation's security operations. As the work is currently at the design and proposal stage, future work will focus on implementing a prototype of the proposed architecture, deploying it within a controlled testbed, and empirically evaluating its detection accuracy, false-positive rate, and impact on attacker dwell time.

A. Future Work

Building on the architecture and evaluation design presented in Sections III and IV, future work will proceed in four stages:

- **Stage 1 — Prototype implementation:** developing the AI Honeytoken Generation module using a large language model API to produce context-aware fake credentials, alongside the Python-based Token Matching Algorithm and Monitoring Engine.
- **Stage 2 — Testbed deployment:** deploying the prototype within a controlled honeypot platform such as Cowrie or T-Pot, following the network segmentation strategy proposed in Section IV-A.
- **Stage 3 — Integration of response tooling:** implementing the Alert Generation, Risk/Event Logging, Database Management, and Security Dashboard modules to complete the end-to-end pipeline.
- **Stage 4 — Empirical evaluation:** measuring detection time, false-positive rate on legitimate traffic,

and attacker dwell time, and comparing these results against the qualitative expectations discussed in Section IV-B.

ACKNOWLEDGMENT

The authors thank the Department of Artificial Intelligence and Data Science for its guidance and support throughout the development of this work.

REFERENCES

- [1] L. Spitzner, "Honeypots: catching the insider threat," in Proc. 19th Annual Computer Security Applications Conf. (ACSAC), 2003, pp. 170–179.
- [2] M. H. Almeshekeh and E. H. Spafford, "Cyber security deception," in *Cyber Deception: Building the Scientific Foundation*, Springer, 2016, pp. 23–50.
- [3] A. Juels and R. L. Rivest, "Honeywords: making password-cracking detectable," in Proc. 2013 ACM SIGSAC Conf. on Computer & Communications Security, 2013, pp. 145–160.
- [4] S. Farooqi, M. B. Musa, Z. Shafiq, and F. Zaffar, "CanaryTrap: detecting data misuse by third-party apps on online social networks," *Proc. Priv. Enhancing Technol.*, vol. 2020, no. 4, pp. 336–354, 2020.
- [5] N. Prabhaker, G. S. Bopche, and M. Arock, "Generation and deployment of honeytokens in relational databases for cyber deception," *Computers & Security*, vol. 146, art. 104032, Nov. 2024.
- [6] D. Reti, K. Elzer, and H. D. Schotten, "Act as a honeytokens generator! An investigation into honeytokens generation with large language models," in Proc. 11th ACM Workshop on Adaptive and Autonomous Cyber Defence (AACD '24), Salt Lake City, UT, USA, Dec. 2024, pp. 1–12.
- [7] M. Sladić, V. Valeros, C. Catania, and S. Garcia, "VellMes: a high-interaction AI-based deception framework," in 2025 IEEE European Symp. on Security and Privacy Workshops (EuroS&PW), 2025, pp. 671–679.
- [8] R. A. Bridges, T. R. Mitchell, M. Muñoz, and T. Henriksson, "SoK: honeypots & LLMs, more than the sum of their parts?," *arXiv preprint arXiv:2510.25939*, 2025.
- [9] T. Yu, Y. Xin, and C. Zhang, "HoneyFactory: container-based comprehensive cyber deception honeynet architecture," *Electronics*, vol. 13, no. 2, art. 361, 2024.
- [10] A. I. Weinberg, "PHANTOM: polymorphic honeytokens adaptation with narrative-tailored organisational mimicry — contextually convincing cyber deception at scale," *arXiv preprint arXiv:2605.02992*, 2026.