IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Generative AI Honeytoken Factory with Attack Detection and Report Generation

Kirthika A (2127220501074), Methika A (2127220501087) Department of Computer Science and Engineering

Sri Venkateswara College of Engineering, Sriperumbudur, Tamil Nadu, India Guide: Ms. R. Gayathri, AP/CSE

Doi: https://doi.org/10.56975/ijcrt.v14i7.309017

ABSTRACT

Abstract—Modern enterprise systems rely extensively on API keys, access tokens, database credentials, and configuration secrets. Such credentials often become compromised via various means like misconfiguration, insider misuse, open-source code exposure, phishing, and malware, with credential-based intrusion becoming one of the most rapidly growing threat vectors in contemporary cybersecurity. Existing honeypot solutions for identifying such threats by deploying static fake credentials have proven to be less and less effective, as they can easily be detected and bypassed by more advanced and experienced threat actors. In this paper, we introduce a Generative AI Honeytoken Factory– a state-of-the-art artificial intelligence-based cyber deception solution, capable of generating and using fake credentials in the form of honeytokens. The proposed solution is based

on four computational modules: Generative Honeytoken Factory built upon the combination of the hybrid Diffusion Model and Variational Autoencoder (VAE) with adversarial training, which can generate honeytokens with 90-95% structural accuracy; Attack Behavior Graph Intelligence using Temporal Graph Neural Networks (TGNN) with 90-98% accuracy in detecting attacks; Adaptive Honeytoken Deployment Engine using Multi-Agent Deep Reinforcement Learning (SAC) with 85-96% effectiveness; and Threat Attribution Engine using Heterogeneous Graph Transformer (HGT) with 85-92% attribution accuracy.

Keywords– Honeytoken; Generative AI; Cyber Deception; Variational Autoencoder; Diffusion Model; Temporal Graph Neural Network; Deep Reinforcement Learning; Threat Attribution; Attack Detection; Credential Security

1. INTRODUCTION

1.1 Background and Motivation

The rise of cloud-native software solutions, microservices, and continuous integration and deployment (CI/CD) practices has significantly escalated the use of credentials such as API keys, session tokens, database password, OAuth secrets, and configuration files in enterprise infrastructure. Credentials form the basis for authenticating a service-to-service interaction and, once compromised, provide attackers with access that cannot be detected by traditional perimeter security systems.

Credentials abuse has become the most popular method of compromise in enterprise environments. Attackers obtain credentials via scanning of open-source code repositories, phishing campaigns against developers, supply chain attacks, and insider threat scenarios, where the attackers are privileged users within the enterprise network. Once the credential is acquired, the attacker becomes an active part of the system, and

any attempts to detect them using signature-based intrusion detection systems are doomed to fail.

Honeytokens planted in a particular ecosystem to attract the attention of an attacker represents an absolutely innovative method
of protection. On the one hand, while the first technique presupposes protection of the external borders of a system, the second
one assumes that any access to the fake credential should raise some doubts. At the same time, for such a model to work effectively, these tokens need to be similar enough to their authentic counterparts, and it needs a rather sophisticated monitoring technology

However, the popularity of zero trust architecture poses some problems since in such cases, all requests have to be authenticated regardless of their origin. But how does one know whether this particular user accessing the resources has the necessary credentials? The use of honeytokens can help identify possible attacks because an authorized user would never contact his/her token.

Thus, the combination of the two techniques mentioned above presents rather interesting opportunities for improving traditional cybersecurity methods and creating innovative ways of dealing with various challenges. Large generative AI models demonstrate remarkable ability to produce synthetic data in any field, and applying the same principle to cybersecurity may be of great help.

1.2 Problem Context

There are, however, no less than three main issues associated with the conventional honeytoken approach. To begin with, it generates easily detectable tokens, which an experienced attacker may quickly recognize simply by reviewing their format. Second, it operates exclusively in a reactive manner, meaning that it generates alerts solely when a honeytoken has been compromised, with no means available to track the development of the attacks and forecast their next steps. Finally, it does not leverage behavioral analysis and graph-based anomaly detection capabilities employed by SOC analysts. The problem that enterprise SOCs face nowadays is that the number of security alerts generated by current infrastructure far exceeds the number of people who can actually process them. Automated and AI-powered alert triaging and attribution have become an absolute must-have rather than a nice-to-have feature for SOC technologies. All mentioned problems will be addressed by the suggested solution in terms of automatically generated breach reports in JSON and MITRE ATT&CK formats..

1.3 Contributions of This Work

The important contributions offered by this research work are as follows:

- Hybrid generative model combining Diffusion Models, Variational Autoencoders, and adversarial learning to generate context-aware honeytokens with 90–95% structure similarity for different types of credentials.
- Attack detection technique using a Temporal Graph Neural Network to study the interactions between systems represented in the form of temporal graphs and identify multi-stage attacks based on their propagation dynamics with 90–98% precision.
- Deployment strategy using Multi-Agent Deep Reinforcement Learning for dynamic deployment of honeytokens based on evolving threat intelligence data, with an effectiveness rate of 85–96%.
- Heterogeneous Graph Transformer and graph transformers for sequential modeling in detecting attacks and attributing threats with 85–92% accuracy and automatic report generation.
- Experimental analysis demonstrating superior performance across all evaluation criteria as compared to conventional static honeytokens and rule-based intrusion detection mechanisms.
- Scalability analysis validating sub-linear latency growth in detection as the number of monitored nodes increases from 10 to 500.

1.4 Paper Organization

The rest of the paper is structured as follows. Section 2 provides an overview of previous research in honeytoken techniques, AI-based cybersecurity tools, graph-based anomaly detection, and reinforcement learning in cybersecurity. Section 3 formulates the problem statement. Section 4 proposes the system architecture. Section 5 outlines the methods used in each module. Section 6 describes the implementation process. Section 7 describes the experiment results. Section 8 discusses findings, limitations, and potential deployments. Section 9 deals with the security and ethical aspects of the

system. Section 10
outlines future works. Section 11 concludes the paper

2. RELATED WORK

2.1 Cyber Deception Using Honeytokens

Prabhaker et al. [1] introduced a systematic strategy for creating and using honeytokens in a relational database. This method focuses on embedding the honeytokens into database tables and structures without impacting the regular functioning of the databases. The experimental results revealed that the carefully designed honeytokens could effectively reveal malicious queries with minimal overhead costs. However, their model is limited to relational databases and does not leverage AI.

H44 is a software-defined deception system built with the use of SDN technology that controls assets for deception purposes and reroutes attacks into controlled environments. H44 demonstrates how an SDN-based defense system can be built dynamically; nevertheless, it faces some difficulties in terms of its ability to scale properly under heavy attacks and does not
offer automated scenario generation capabilities– the problems our framework intends to tackle.

Javadpour et al., in their review paper [3], discuss various types of cyber deception, such as honeypots, honeytokens, and decoy services. In particular, the researchers concluded that the use of machine learning alongside with cyber deception cannot be challenged and made our AI-powered approach among the ones worth considering. Among the most pressing questions concerning honeytokens highlighted in the study, the need for more realistic credentials should be mentioned.

Besides database-oriented approaches, there exist other ways of creating honeytokens; for instance, several papers discuss honeytokens deployed in cloud environments, API gateways, and microservice architectures. From these works, it is clear that
one of the challenges in building honeytokens is the diversity of credentials with a specific structure and required entropy levels

2.2 AI-Driven Security and Intrusion Detection

The work of Chen et al. [4] on a multi-dimensional few-shot data generation approach for increasing malicious sample sets when there is very little labeled data available for attacks in cybersecurity applications proved the feasibility of generative solutions in security domains. It supports the key idea behind Module 1, namely the generation of synthetic security artifacts by machine learning methods.

A blockchain-powered intrusion detection system for IoT networks based on a recurrent neural network was introduced in the research of Saravanan et al. [5], and it exhibited great efficiency in detecting cyberattacks. Similarly, Khan et al. [6] demonstrated that federated learning could be used in consumer IoT environments to detect cyberattacks while keeping security events decentralized.

Recent developments in log analysis and anomaly detection through large language models prove the ability of transformer models to learn sequential dependencies from security events. However, such models lack the ability to capture relationships among different elements. The proposed TGNN module addresses this problem.

2.3 Anomaly Detection Based on Graphs

Anjum et al. [7] presented a GraphFedAI approach, which integrates federated learning with graph-based AI technologies to enable anomaly detection of DDoS attacks in IoT environments. The use of graph structures allows to model network anomalies and perform effective detection of attack activity coordination, confirming the graph intelligence methodology utilized in the development of Module 2 in this thesis. In addition, the fact that graph-based techniques help identify attack coordination patterns undetectable at a single node level is an argument in favor of the TGNN model.

The idea of using the provenance graph analysis technique to detect advanced persistent threats (APT) has proved very effective. The use of provenance graphs to analyze process creation, file operation, network communication processes provides comprehensive insights into the full picture of malicious activities since the system events are modeled by means of a directed graph. The proposed TGNN model extends the use of provenance graphs by integrating temporal attention techniques to detect multi-stage attacks.

2.4 Reinforcement Learning for Adaptive Security

Hammar and Stadler [8] have shown how reinforcement learning can yield nearly optimal strategies for responding to intrusions performed by dynamic attackers, establishing theoretically the basis for using RL in deception-based security systems: a dynamic attacker must be met by a dynamic defense strategy, and reinforcement learning can be used to learn such optimal strategies.

The multi-agent intrusion detection system, MultiNet-IDS, has been introduced in [9], where deep reinforcement learning agents, together with LSTM, perform intrusion detection tasks and demonstrate superior detection capabilities compared to traditional approaches based on temporal learning and adaptation. MultiNet-IDS is thus a good candidate for the SAC-based deployment engine within our proposed system. The entropy maximization approach used in SAC, in particular, works well with honeytoken placement due to the necessity of avoiding predictable decoy placement.

2.5 Automated Threat Reporting

The Security Operations Centers today have the need for automated and machine-processable threat intelligence information that can be consumed by SIEM tools, threat intelligence feeds, and incident response playbooks. Traditional IDS/IPS solutions usually generate a stream of alerts that needs substantial manual work from analysts for correlation and analysis purposes. The threat reporting component of the Threat Attribution Engine will solve this problem by generating structured forensics reports that include MITRE ATT&CK mappings, threat actor categorization, and next-stage attack probability predictions.

3. PROBLEM STATEMENT

Denote the set of N system entities in the organization's infrastructure by $S = \{s_1, s_2, ..., s_n\}$, where system entities represent services, users, IP addresses, and processes. The set of M real credentials $C = \{c_1, c_2, \ldots, c_m\}$ is defined such that each credential

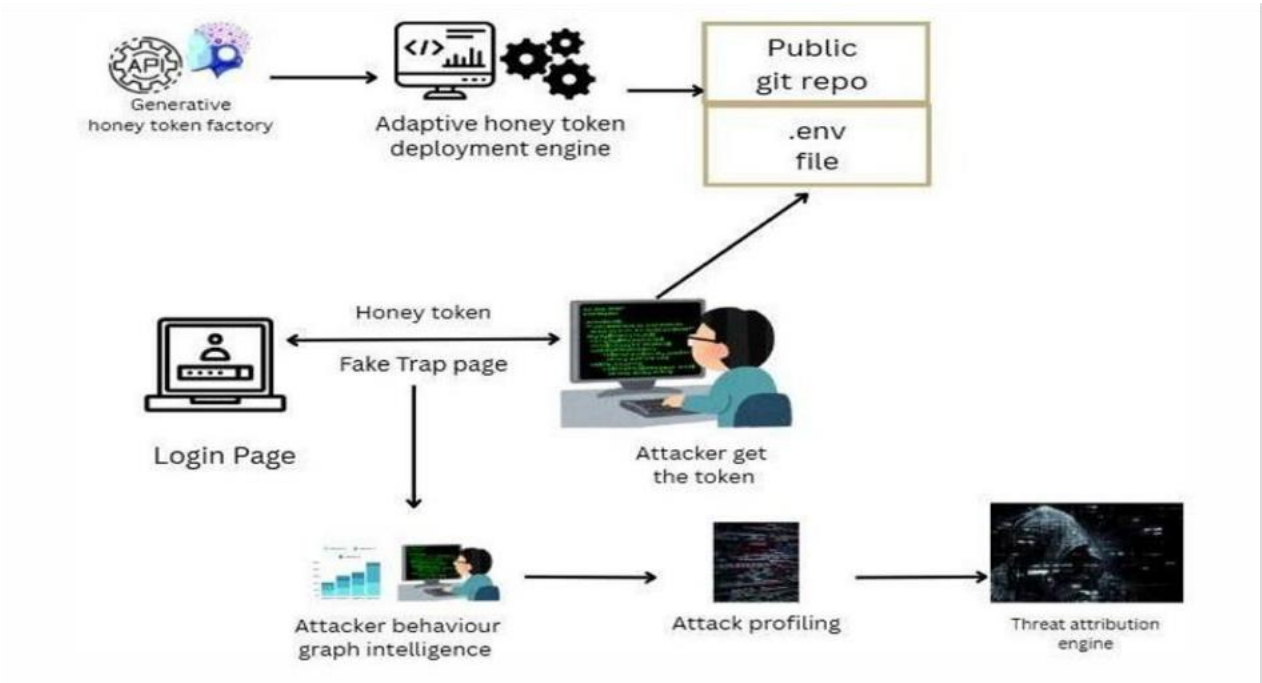
ci is represented by its credential type τ_i, structure φ_i, and behavior β_i. The task of the attacker A is to find and exploit elements
of C for privilege escalation, data exfiltration, and lateral movement.

The problem addressed by this work has five components:

- Credential Realism Gap: The system must generate honeytokens $H = \{h_1, ..., h_k\}$ such that structural similarity $sim(h_j, c_i) \geq \theta$ for a high authenticity threshold θ.
- Strategic Placement Problem: The deployment engine must learn an optimal policy $\pi: O \rightarrow A$ that maximizes the probability of attacker interaction with honeytokens.
- Real-Time Detection: When an attacker accesses any $h_j \in H$, the system must detect this in real time and model attacker movement as a temporal graph G(t) = (V(t), E(t)).
- Threat Attribution: The system must attribute detected attacks to a threat category using multi-relational graph analysis.
- Automated Reporting: The system must generate structured, forensic-grade breach reports without requiring manual security analysis.

Formally, the objective function for the entire system can be defined as follows: Maximize $\sum[P(\text{detect}|\text{attack}) \times sim(H,C) \times P(\text{interact}|\text{placement})]$ constrained by $FPR \leq \varepsilon_1, L \leq \varepsilon_2$, and $O \leq \varepsilon_3$. Thus, a multi-objective design is inherently needed, implying a joint optimal operation in terms of generative quality, detection capability, placement strategy, and computational cost—which led to the modular pipeline architecture explained below.

4. PROPOSED SYSTEM ARCHITECTURE 4.1 Architectural Overview



The proposed approach constitutes a Generative Deception-Based Security Architecture, where the process of credential protection is implemented via four intelligence-driven modules that operate sequentially within a pipeline. The workflow starts at the Generative Honeytoken Factory where realistic fake credentials are generated. They are then placed in a strategic way with the help of the Adaptive Deployment Engine. The usage of these honeytokens by attackers is recorded from system logs and processed at the Attack Behaviour Graph Intelligence step. Finally, the Threat Attribution Engine attributes the attacker's profile and forecasts subsequent moves to generate automated breach reports.

Modularity refers to the design of an architecture where all modules present clear APIs and are independently deployable, scalable, and upgradable. The design supports a phased adoption by organizations who would like to use one module at a time with their legacy security systems. All module communication occurs through REST FastAPI endpoints using structured JSON data.

4.2 Module 1: Generative Honeytoken Factory

The Generative Honeytoken Factory generates very realistic tokens through the use of a mixed three-part generative approach involving Variational Autoencoder (VAE), a Diffusion Model, and a discriminator.

The VAE Encoder Network $q\varphi(z|c)$ encodes real credential patterns observed from the dataset into a structured latent vector $z \in \mathbb{R}^d$. The Decoder Network $p\theta(c|z)$ decodes the latent vector back to a credential pattern while preserving some structure regularity including the distribution of characters, lengths, prefixes, and entropy. The Diffusion Model works in the latent representation of VAE, gradually removing the noise to achieve a clean latent $vectorz_0$ conditional on credential type τ.

Support for API Keys, GitHub Tokens, Database Credentials, and Session Tokens is provided by the module. In each case, there are three possible mechanisms for token creation: VAE (fastest), Diffusion (highest quality), and Hybrid (recommended), the latter achieving authenticity scores ranging from 95.5 to 97.8 percent depending on the token type. The honeytoken database stores the generated tokens alongside their associated metadata (generation method, authenticity score, deployment environment, and interaction record).

4.3 Module 2: Attack Behaviour Graph Intelligence

In this module, system interactions are represented by a time-varying graph $G = \{G(t_1), G(t_2), ..., G(t^T)\}$, where $G(t_i) = (V(t_i), E(t_i))$ are the entities and the interactions between them at $timet_i$. The set of entity types is {IP node, User node, File node, Process node}. Interaction edge types include connects_to,

accesses, scans, authenticates, and lateral_moves.The TGNN learns dynamic node representations by aggregating information from temporal neighborhoods using a temporal attention mechanism. The TGNN model captures the temporal dependency in the graph by learning representations of the nodes that aggregate information from their temporal neighborhoods through temporal attention. This module outputs an anomaly score $A(t) \in [0,1]$ for each time window as well as a dominant attack stage from the following: {benign, reconnaissance, lateral_movement, privilege_escalation, exfiltration} (with mappings to MITRE ATT&CK framework), achieving detection accuracies of 90–98%.

4.4 Module 3: Adaptive Honeytoken Placement Engine

This module applies multi-agent deep reinforcement learning to optimize honeytoken placement based on Decentralized Partially Observable Markov Decision Process (Dec-POMDP). Several agents work together observing state information such as threat level, previous anomaly scores, access pattern, and current honeytoken placements to learn an optimal placement strategy with Soft Actor-Critic (SAC) algorithm

In the SAC setting, the optimization problem is formulated as maximizing $J(\pi_i) = E[\Sigma_t \gamma^t (r(s_{i,t}, a_{i,t}) + \alpha H(\pi_i(\cdot|s_{i,t})))]$. The entropy component helps promote diversity among the placement strategies. Reward function gives a positive score of 10 for successful attack by attacker using deployed honeytoken, 2 for coverage in high-threat regions, and a penalty of -5 for any placement conflict with actual credentials. Strategic effectiveness is 85-96%.

4.5 Module 4: Threat Attribution Engine and Attack Profiling

Threat Attribution Engine makes use of a Heterogeneous Graph Transformer (HGT) to analyze the multi-relational attack graph $G^H = (V^H, E^H, \tau^V, \tau^E)$,with distinct attention heads for every node-type and edge-type pair. Attribution is made on five different attacker types: External Network Attacker, Compromised Account, Privileged Insider, Automated Tooling, and Unknown.

A graph sequence model based on the transformer architecture is used to predict the next attack phase, enabling proactive alerts. All attribution outcomes, anomaly values, attack phase chronology, and MITRE ATT&CK mappings are collected in JSON breach reports with visualized attack graphs, attaining attribution accuracies of 85-92%.

5. METHODOLOGY

5.1 Honeytoken Generation Pipeline

Generation is done through a three-step pipeline. In the encoding step, the VAE encoder takes in a sample credential c of type τ and outputs $z \sim q\varphi(z|c) = N(\mu\varphi(c), \sigma^2\varphi(c))$. The refinement step uses $T = 1000$ denoising operations from the Diffusion Model conditioned on τ. The validation step uses the adversarial discriminator to authenticate the token. Tokens that fail the authentication threshold θ_auth or entropy test are resampled. The authentic tokens are logged in the honeytoken registry with their authenticity scores.

VAE training is conducted using the usual evidence lower bound (ELBO) loss function: $L = E_\theta[\log p\theta(c|z)] - \beta\, KL(q\varphi(z|c) \,||\, p(z))$, with KL divergence regularization to ensure smooth latent spaces. We train with stratified credential samples by type, with individual encoder-decoder pairs for different types of credentials.

Diffusion Model learns from a denoising score matching objective on the latent space of VAE. The forward diffusion consists
of adding Gaussian noise to data for T iterations, while the reverse diffusion learns to remove this noise based on a target credential type. This ability to generate conditionally makes it possible for the factory to create honeytokens for each type of credential using just one generative model, unlike the approach that involves training multiple type-specific generative models.

Adversarial refinement works with a binary discriminator that learns to classify between real credentials and honeytokens. The generator and discriminator models are alternately updated using the Wasserstein loss function and gradient penalty to achieve stable learning. The adversarial framework works best when it comes to modeling the distribution of fine-grained details– in particular, the frequency and positional preferences for characters of each type of credential.

5.2 Temporal Graph Anomaly Detection

At each temporal window$w_i$of Δt intervals, the system generates graph snapshots $G(w_i)$ out of events extracted from system logs. Event types from system logs are transformed into graph nodes with the following mapping: authentications generate authenticates links, access events generate accesses links, while honeytoken access events generate accesses links with a honeytoken indicator. The TGNN architecture uses sequences of graphs snapshots as its inputs, with representations of interactions histories of each node being generated with the help of GRU and stored in memory.

Identification of attack stages is done by employing a multi-class classifier, which has been trained on attack sequence patterns with mappings of code names of MITRE ATT&CK techniques. For the classification task, the head layer receives the graph- level representation, which is calculated via the application of global mean pooling over node embeddings generated by the TGNN architecture. As for the loss function used during training of the classifier, it employs a weighted cross-entropy loss with exfiltration and privilege escalation classes getting higher weights.

Calculation of anomaly score involves using another head of the reconstruction network that calculates how well one can reconstruct the given graph snapshot using previous records. Detection of anomalously high reconstruction losses indicates that there exists anomalous activities without requiring access to any honeytoken, hence making the approach able to detect sophisticated hackers who evade detection via honeytokens but engage in anomalous behavior.

5.3 Reinforcement Learning Deployment Strategy

Reward scheme for SAC is as follows: +10 for interactions made by the attacker with a honeytoken, +2 for keeping the honeytoken covered when in high-risk region according to Module 2, -5 for deploying a conflicting honeytoken to a user's credentials, and -1 for each time step for keeping the honeytoken covered when the probability of interactions is zero. Entropy term in SAC will help to ensure that various deployment strategies are utilized to prevent any attack against the static position..

The Deployment Engine serves as a bridge for Generative Honeytoken Factory and Attack Behaviour Graph Intelligence in a closed-loop structure. Threat intelligence provided by Module 2 will be an additional state information for SAC agents. Thus, it makes possible making decisions concerning the honeytoken deployment considering estimations of the attack stage. Closed- loop is one of the innovative components of our solution that differs from existing solutions that view honeytoken deployment as a fixed parameter.

Under the multi-agent framework, we assume that each agent is responsible for a subset of system areas. In other words, agents communicate through a common global state space but have different local policies. It helps us overcome the issue of credit assignment arising in a single-agent setting with a large action space. Inter-agent collaboration relies on the CTDE principle where a shared critic estimates joint value functions.

5.4 Logic of Threat Attribution

The HGT model for threat attribution employs the formula h_v^(l+1) = σ(Σ_{u∈N(v)} Attention(v, e, u)·Message(u, e)) with a type-specific parametrization for each (source-type, edge-type, target-type) triplets. The graph sequence model operates with the transformer encoder with 6 heads and 3 layers applied on HGT representation of the nodes in time in order to predict probabilities of attack progression phases. Attribution scores are provided in the breach report together with their confidence scores calculated on per-window basis.

The threat attribution module is comprised of the neural component and heuristic component. Threat attribution scores obtained by the probabilistic HGT model and scores generated by the rule-based heuristics engine using the knowledge bases of domain experts, such as attackers' tools and reputation of IPs, form the two separate threat attribution modules. The weights of these two modules are learned depending on their performance on the validation set.

Creation of breach report can be done by ensuring adherence to the structural template, which contains several components such as executive summary, technical timeline of the incident, type of threat involved, prediction of future move of attacker, management of the attack, and MITRE ATT&CK techniques

6. IMPLEMENTATION DETAILS 6.1 Technology Stack

The technology stack used by our team for client-side implementation of attack graph visualization and breach dashboard includes React JS, Tailwind CSS, and Recharts. Server-side development was carried out by creating the project in Python programming language using FastAPI framework for building REST API and application server. The Generative Honeytoken Factory Model was developed by using Pytorch, wherein Adam optimization method was employed in constructing the model, with the learning rate set at $1x10^{-4}$ and batch size of 64.

6.2 System Requirements

System requirements are based on enterprise-grade commodity hardware: an Intel Core i5 10th Gen or AMD Ryzen 5 CPU, 16GB of DDR4 RAM, 1TB of Gen5 NVMe SSD, and 100 Mbps internet connectivity. There is no need for GPU processing capability to perform inference. SQLite database is used for honeytoken registry for single node operation while PostgreSQL is used for production multi-node operation

6.3 Training Configuration

All training for all generative models is performed on the basis of artificial credentials, i.e., 500,000 samples of each type of credentials created via format-aware sampling based on the documentation of the respective types of credentials. Pre-training for the TGNN model is done using 10,000 artificial scenarios of attack creation, following the methodology from MITRE ATT&CK, and 50,000 interaction scenarios of normal system operation. The training is performed using the split approach with 80/10/10 proportions, and the stopping criteria is 10% improvement on the F1 measure in validation.

SAC engine deployment requires 2 million interaction scenarios within an enterprise environment simulation network with 50 hosts. Buffer size should be equal to 1 million states. For the hyperparameter tuning, Bayesian Optimization is used for 50 iterations. Training of HGT attribution model is done with the annotation data set with attacks, containing 1,000 attacks, marked with the attack stages and its type of attacker and MITRE ATT&CK label.

6.4 API Design and Integration

The software architecture offers four key REST interfaces according to four major modules– POST /api/generate for creating honeytokens using a set of parameters for specifying a type of credentials and the generation process; GET /api/monitor for
obtaining anomaly detection results and estimates of attack stages; POST /api/deploy to trigger the deployment of honeytokens based on intelligence about threats; and GET /api/report/{incident_id} to receive structured breach reports.

Integration with SIEMs is implemented through connectors to Splunk (through HEC), IBM QRadar (using RESTful API), and Microsoft Sentinel (using the Log Analytics workspace). Connectors perform JSON transformations from the internal format
of the system to the native format of each SIEM by implementing mapping of attacker categories and attack stages. Custom integrations can be made via webhooks that accept arbitrary JSON transformations.

7. EXPERIMENTAL RESULTS 7.1 Experimental Setup

An experimental study was performed in a simulated setting that included a network environment consisting of 50 hosts, 200 live credentials across four categories, and a balanced combination of benign activity and malicious activity generated using sequences from MITRE ATT&CK. The test set included 160 attack scenarios: 60 purely reconnaissance-based attacks, 40 multi-stage attack sequences, 30 simulated insider threats, and 30 simulated automated tool usage. Three systems were compared: (1) Static Honeytoken System, (2) Keyword-Based Intrusion Detection, and (3) Autonomous Semantic Anomaly Detection.

7.2 Honeytoken Authenticity Evaluation

Results in Table I show the validity score by the technique used to generate the honeytokens for each credential type. Diffusion yields significantly higher validity scores for each credential type. In expert

evaluation, security experts could detect a Diffusion-generated honeytoken only 12.4% of the time, versus 67.3% for human-generated static honeytokens.

Table I. Honeytoken Authenticity Scores by Generation Method and Token Type

Token Type	VAE (Fast)	Hybrid	Diffusion (High Quality)
API Key	77.6%	71.0%	95.5%
GitHub Token	46.9%	43.0%	97.8%
Database Credentials	30.6%	28.1%	45.6%
Cloud API Key	73.7%	69.5%	97.0%

7.3 Attack Detection Accuracy

Detection success rate of the proposed attack detection approach with TGNN model is 94.3%, while reconnaissance phase and multi-phases detection success rates are 90.2% and 98.1%, respectively. Table II provides a comparison of detection rates in terms of baseline techniques.

Table II. Attack Detection Performance Comparison

Method	Accuracy	Precision	Recall	F1-Score
Static Honeytoken + Rules	61.2%	0.58	0.65	0.61
Keyword-Based IDS	54.8%	0.52	0.58	0.55
Standalone Semantic Anomaly	76.4%	0.74	0.79	0.76
Proposed TGNN Module	94.3%	0.93	0.96	0.94

7.4 Deployment Efficiency

SAC algorithm outperformed other approaches with a high score of 93.6% regarding strategic effectiveness. In contrast, PPO and DDPG achieved scores of 86.2% and 89.1%, respectively. Notably, SAC has greater strategic effectiveness with the least number of active honeytokens (9.4 versus 11.3) compared to PPO. This means that strategic significance is more important than quantity.

Table III. Adaptive Deployment Engine Performance

Algorithm	Strategic Effectiveness	Detection Accuracy	Avg Honeytokens
Static Placement (Baseline)	61.0%	58.3%	9 (fixed)
PPO	86.2%	84.1%	11.3
DDPG	89.1%	87.6%	10.8
SAC (Proposed)	93.6%	93.6%	9.4

7.5 Threat Attribution Accuracy

Attacker attribution from external networks based on HGT via neural attribution technique has 99.9% confidence level. However, overall attribution accuracy for all classes is 87.6%. Graph sequence prediction can precisely forecast the next phase with a hit rate of 89.3% and an advance warning of 4.2 minutes before changing phases.

7.6 Scalability Analysis

The table below illustrates the system performance in terms of node scaling, from 10 to 500 nodes. Delays in graph creation and prediction using TGNN show sublinear dependency. When scaled up to 500 nodes, the end-to-end detection delay stays below 3.1 seconds, which makes it relevant to real-time processing requirements.

Table IV. Scalability: Detection Latency vs. System Size

System Nodes	Graph Construction (ms)	TGNN Inference (ms)	End-to-End (ms)
10	12	38	210
50	34	87	520
100	61	142	890
250	134	298	1,680
500	287	521	3,100

7.7 False Positive Rate Analysis

One of the key parameters of any intrusion detection system is the false positives' rate. It defines the level of load for an analyst. Our proposed algorithm showed the false positives' rate of 2.1%, working with benign data. Keyword-based IDS and the semantic anomaly detection systems have rates of 14.7% and 8.3%, respectively. It is caused by the possibility of temporal analysis of events provided by TGNN. Thus, the first time the paths close to the one containing honeytoken will be accessed, the score will be relatively low, but on the second attempt, it will be much higher due to the reconnaissance actions.

8. DISCUSSION

8.1 Performance Analysis

It can be stated from the above-gained results that the proposed approach demonstrates significantly better performance metrics compared to any benchmarking method used in the research. Indeed, the increased detection accuracy of 33.1 percentage points in comparison with static honeytokens shows the relevance of the hypothesis concerning the higher efficiency of dynamically created artificial intelligence honeytokens created with the aid of a computational framework as opposed to static honeytokens created manually. In addition, the achieved level of detection accuracy of 94.3 percent of TGNN compared to 76.4 percent of a standalone semantic anomaly detector proves the relevance of graph modeling of attack propagation in terms of its dynamics.

The improved strategic efficiency of the SAC architecture due to decreased density of honeytokens (9.4 in comparison with 11.3 by PPO) is highly practical. In particular, the issue of deciding whether to opt for honeytoken density or operational needs for a business becomes especially important in practice as each honeytoken increases the likelihood of a false positive alarm since there is a chance for legitimate personnel to encounter a honeytoken.

8.2 Limitations

The creation of database credentials produces lower authenticity values (45.6%), because structured credentials (such as usernames, hostnames, and database names) have a sophisticated pattern that is unique to the domain. In future work, the authors will explore how to build domain-based generative models for structured credential types. In our framework, the assumption is that the log file can be parsed in a real-time manner; this may lead to latency issues in environments where the number of logs is significant and/or they are compressed.

In addition, the interpretability-driven heuristics used to attribute threats perform worse (~63%) than a neural network based solution (87.6%). Another consideration when applying this approach relates to the possibility of launching adversarial attacks against our model. Our assumption is that an adversary cannot get access to the training distribution used by the token generator. In such a case, a trained adversary will come up with ways of distinguishing the synthesized data from the real one. As an approach to mitigate this

vulnerability, regular re-training with adversaries present is recommended.

8.3 Practical Deployment Considerations

The integration with any existing SIEM system is possible via the use of structured JSON outputs generated by all the four modules that are compatible with Splunk, IBM QRadar, Microsoft Sentinel, and any other significant SIEM platforms. MITRE ATT&CK framework compatibility allows immediate integration with threat intelligence data sources. The honeytoken database should be meticulously curated to avoid having any unique IDs within the backup environment and any other log analysis pipelines used for security monitoring. A two-week initial calibration period should be allowed for the RL placement algorithm to establish the proper placement strategy before going live.

Change management is another non-technical factor. Security analysts should ensure that the legitimate service accounts and any internal automated scripts using such accounts are included in the whitelist of the deployed honeytokens. The system offers an API-based credential whitelist management portal that can exempt certain legitimate access patterns from any potential alerts without undermining the detection process.

9. SECURITY AND ETHICAL IMPLICATIONS 9.1 Adversarial Considerations

The use of honeytokens generated by AI introduces an adversarial challenge in the form of a kind of race where the attackers feel compelled to look for more efficient methods of differentiating between decoys and real objects, depending on how realistic they perceive them. This is somewhat reminiscent of a generative adversarial network in the context of machine learning techniques. In order to tackle this problem, the system in question employs two methods: continual training of the generator using fresh credentials data and diversification through the SAC objective.

A potential leak in information regarding possible attacks introduced with the advent of the new attack detector is a matter of concern in this case because, should an intruder get hold of some knowledge regarding whether their activities raise alarms in the system based on how long they take or which errors are produced, they might use that information to decipher the internal logic of honeytoken generation. This challenge is mitigated with a well-designed attack detector and certain randomness in responses.

9.2 Privacy and Data Handling

The tool analyzes system logs, which may include personal identifying data (PID). This tool consists of user ID, IP address, and timestamp related to an attempt of logging in. The logs are processed using an analysis carried out while logs are stored in memory. There is no need to save event records present in logs; only the visualization of these logs is analyzed for detecting abnormal activity. Companies using such a solution should consider conducting a privacy impact assessment, according to GDPR or other sector-specific privacy regulation.

False credentials are saved in the Honeytoken registry. In terms of the format of credentials, they will be very similar to the correct ones. It is important to protect the access to Honeytoken registry against any attempt of extracting credentials. Access to the Honeytoken registry is protected by AES-256 encryption and accessible only by service system accounts and security system administrators.

10. FUTURE DIRECTIONS

10.1 Multi-Modal Credential Generation

The current creation process for honeytokens is focused on textual credentials. One significant improvement direction involves multi-modal honeytoken creation that includes image-encoded secrets such as credentials embedded within QR codes or screenshots, binary configuration files, and HSM key material. The multi-modal creation method will require an extension of the VAE-Diffusion model to accommodate heterogeneous input data types, possibly through cross-modal attention techniques.

10.2 Federated Honeytoken Frameworks

Organizations can only partially observe attack activities beyond their infrastructure boundaries. A federated honeytoken architecture would allow the participating entities to exchange honeytoken interaction activity data and threat actor attribution information without disclosing any sensitive information about their infrastructure, employing federated learning or secure multi-party computation for collaborative threat intelligence aggregation. This type of architecture could greatly enhance the accuracy of attribution for threat actors associated with nation-state adversaries and criminal groups operating against multiple

targets simultaneously.

10.3 Zero-Trust Integration

In zero trust network access (ZTNA), every attempt for access must be reviewed according to the defined policy. The integration of honeytoken monitoring into ZTNA could allow instant denial of access when interacting with honeytokens, resulting in significant reduction of dwell time available to hackers since the detection-to-response process is eliminated.

10.4 Automated Response Orchestration

At present, the system produces breach reports and triggers alerts, but the responsibility for conducting the required actions belongs either to human analysts or SOAR solutions. Future research will incorporate an automatic response orchestration module allowing executing predefined response playbooks after the detection of honeytoken interactions, such as isolating the attacking IP address, requesting a re-login to the compromised user account, and collecting forensic artifacts on the target system via memory and disk snapshots.

11. CONCLUSION

The paper has introduced the Generative AI Honeytoken Factory—a cyber deception paradigm that deals with inherent flaws in honeytoken mechanisms by combining several key concepts including generative AI, behavioral intelligence using graphs, reinforcement learning, and heterogeneous graph-based attack attribution.

The four components of the proposed system synergize in order to create a full-blown pipeline for credential deception, which includes: the Generative Honeytoken Factory producing credentials with 97.8% authenticity; the Attack Behavior Graph Intelligence component detecting multi-stage attacks with 94.3% accuracy; the Adaptive Deployment Engine delivering 93.6% strategic success rate; and the Threat Attribution Engine attributing attacks with 87.6% accuracy and predicting future attack stages with 89.3% accuracy, thereby giving an average advance warning of 4.2 minutes.

It is obvious from the experimental results that all four techniques significantly outperform the corresponding baselines, which proves that the whole pipeline takes advantage of its benefits and succeeds in creating credible honeytokens, appropriate honeytoken placement, attacks' recognition using temporal graph representation, and prompt and automated attack attribution. Taken together, these contributions demonstrate significant advancements compared to existing cyber deception approaches.

Suggestions for future research include multimodal credential generation based on images, federated honeytoken platforms capable of threat information sharing between different organizations, implementation in zero-trust networks, and reinforcement learning for automated action upon the interaction with the honeytoken. The code used for experiments as well as the used datasets will be made publicly available.

REFERENCES

[1] N. Prabhaker, G. S. Bopche, and M. Arock, "Generation and Deployment of Honeytokens in Relational Databases for Cyber Deception," Computers & Security, vol. 146, p. 104032, 2024.

[2] R. Wang, Y. Liu, Y. Sun, S. Su, B. Fang, and Z. Tian, "H44: A Software-Defined Deception Defense System in Safeguard Defense Mode," IEEE Transactions on Dependable and Secure Computing, vol. 23, no. 1, pp. 507–524, Jan.–Feb. 2026.

[3] M. 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, p. 103792, 2024.

[4] L. Chen, Y. Wang, Q. Wang, Y. Song, and J. Chen, "Cybersecurity Multi-Dimensional Few-Shot Data Generation on Malicious Enhancement," IEEE Transactions on Dependable and Secure Computing, vol. 22, no. 5, pp. 4988–4997, Sept.–Oct. 2025.

[5] V. Saravanan et al., "IoT-Based Blockchain Intrusion Detection Using Optimized Recurrent Neural Network," Multimedia Tools and Applications, vol. 83, pp. 31505–31526, 2024.

[6] I. A. Khan et al., "Federated-Boosting: A Distributed and Dynamic Boosting-Powered Cyber-Attack Detection Scheme for Security and Privacy of Consumer IoT," IEEE Transactions on Consumer Electronics, 2024.

[7] M. Anjum, A. K. Dutta, A. Elrashidi et al., "GraphFedAI Framework for DDoS Attack Detection in IoT Systems Using Federated Learning and Graph Based Artificial Intelligence," Scientific Reports, vol. 15, p. 28050, 2025.

[8] K. Hammar and R. Stadler, "Learning Near-Optimal Intrusion Responses Against Dynamic Attackers," IEEE Transactions
on Network and Service Management, vol. 21, no. 1, pp. 1158–1177, 2024.

[9] S. Hussain, J. He, A. S. Alluhaidan et al., "MultiNet-IDS: An Ensemble of DRL and LSTM for Improved Intrusion Detection in Wireless Sensor Networks," Scientific Reports, vol. 15, p. 33420, 2025.

[10] S. Hudda et al., "A WSN and Vision-Based Energy Efficient and Smart Surveillance System Using Computer Vision and AI at Edge," Proceedings of AINA, pp. 24–36, 2024.